Ottobre 1984 - Lire 3.000 - Anno I° - N. 4 - Sped. Abb. post. gr. III/70 - Distr. MePe

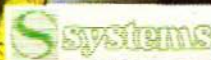
COMMODORE

MENSILE PER UTENT

CONVERSIONI

◆

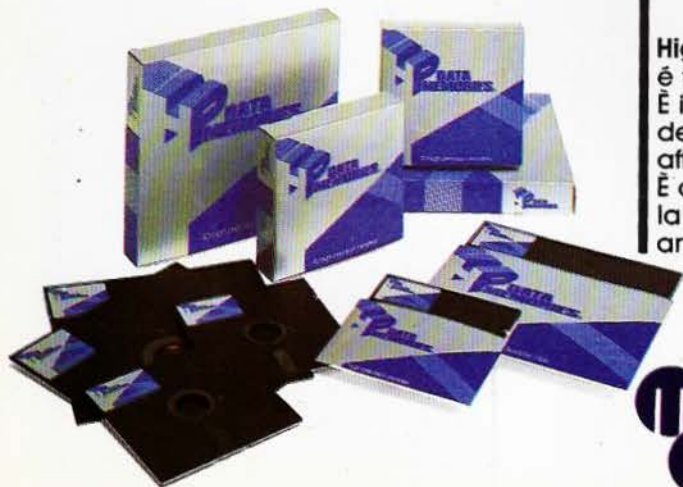
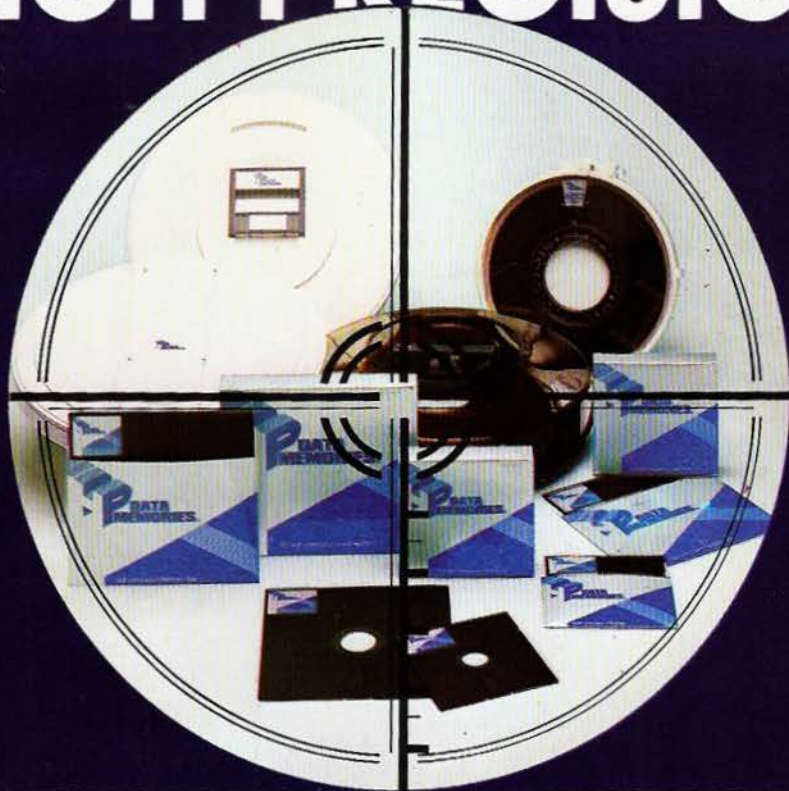
CORNUCOPIA

◆

LO SCOMPATTATORE

◆

UN CORSO
A PUNTATE
DI PROGRAMMAZIONE
STRUTTURATA

PROGRAMMI VALIDI
E TESTATI ANCHE PER IL
C16 E IL PLUS-4

systems

High precision Data Memories
é tecnologia avanzata di costruzione.
È il supporto magnetico testato ai limiti della resistenza con garanzia di assoluta affidabilità.
È avanguardia tecnologica per assicurare la massima protezione dei dati, anche, nelle situazioni più critiche.

HIGH PRECISION A COLPO SICURO!

MEE - Memorie per Elaboratori Elettronici S.p.A.
Forniture per Centri Elaborazione Dati
Sede Amm.va: 20144 Milano - Via Boni 29
Tel. 4988541 (4 linee r.a.) - Telex 324426 MEE-I
Filiali e Agenzie: Milano - Bergamo - Torino
Biella - Padova - Parma - Bologna - Firenze - Ancona
Roma - Napoli - Catania - Oristano - Bari - Genova
Bolzano - Mestre

COMMODORE

DIRETTORE RESPONSABILE
Agostina Ronchetti

REDATTORE CAPO
Gloriano Rossi

REDAZIONE
Eugenio Coppari; Marco De Martino

SEGRETARIA DI REDAZIONE
Maura Ceccaroli

COLLABORATORI
Giancarlo De Cobelli; Marco De Rosa; Valerio Ferri; Francesco Gatti; Mariangela Guardione; Giulio Marcozzi; Mauro Massetti; Ernesto Sidoti; Renzo Zonin.

GRAFICA e IMPAGINAZIONE
Villa Iris s.n.c.
P.zza Massari, 8 - Milano

FOTO
Franco Vignati

DIFFUSIONE E ABBONAMENTI
Marina Vantini

DIREZIONE, REDAZIONE
Viale Famagosta, 75
20142 Milano - Tel. 02/8466675
Autorizzazione del Tribunale di Milano n. 103 del 25/2/84

STAMPA
Litografica - Busto Arsizio

Concessionario esclusivo per la diffusione - MEPE spa
Via G. Carcano, 32 - Milano

Spedizione in abbonamento postale - Gruppo III/70

Prezzo della rivista L. 3.000
Numero arretrato L. 6.000

Abbonamento annuo L. 25.000
I versamenti vanno indirizzati a: Commodore C.C.
V.le Famagosta, 75 - 20142 Milano, mediante emissione di assegno bancario, utilizzando il c/c postale n.ro 31532203

Per i cambi di indirizzo, indicare, oltre naturalmente il nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500 anche in francobolli.

LA POSTA

● **Premetto che sono un vostro abbonato, e tralasciando i soliti complimenti e apprezzamenti di rito, che condivido in pieno, vengo al nocciolo della questione. Sono un utente Commodore, posseggo un C64 e mi diletto nel compilare programmi in basic. Ed ecco il punto: da quando sono entrato in possesso del Simon's Basic ho potuto accertare e vedere la potenza di questo home, fino a quel momento seminascosto fra le migliaia di peek e poke, e soprattutto la facilità di cui lo si può programmare. A parte tutte le istruzioni che facilitano la stesura dei programmi, la programmazione strutturata, la manipolazione dello schermo, ecc. sono stato affascinato dalla velocità incredibile e soprattutto dalla facilità estrema con cui si può disegnare in alta risoluzione disponendo di comandi semplicissimi rapidi e precisi, per non parlare poi della gestione degli sprites che con il Simons è un gioco da bambini. Suonare in Simons' Basic poi è una cosa veramente facile, niente più calcoli niente più insuccessi. Questa stringata lettera mi auguro raggiunga lo scopo di promuovere la diffusione del Simons' Basic perchè credo che l'accoppiata C64 più Simons' renda il C64 l'home più potente, più versatile, e soprattutto l'home più facile da usare. Puntualizzo inoltre che il Simons' è diffusissimo in cassetta e dischi e copiabile, come altrettanto facilmente si trovano le istruzioni. Invito pertanto anche la redazione a tener conto di questi suggerimenti e a pubblicare listati in Simons' Basic. Per chiudere vorrei, se possibile, sapere a che indirizzo decimale inizia la pagina grafica con il Simons' e se è possibile trasferire su disco un'immagine in alta risoluzione creata con il Simons' e se lo è, come farlo.** *(Rosignoli Bruno - Via Tevere, 6 - 06089 Torgiano PG).*

□ Indubbiamente il Simons' Basic è uno strumento veramente eccezionale da utilizzare con il C64. Versatilità, facilità d'uso, potenza sono le sue performances. Sulla nostra rivista presto si inizierà a parlare dei comandi tipo Simons' in quanto il C16 ed il Plus 4 ne prevedono alcuni tipi in comune. Escludiamo ad esempio la gestione degli sprites dato che i nuovi prodotti Commodore ne sono sprovvisti. E non solo, anche i comandi di sound sono limitati, dato che C16 e Plus 4 hanno una generazione di suoni come quelli che troviamo sul Vic 20. Comandi tipo CIRCLE, DROW, BOX. etc. sono uguali o estremamente simili anche come sintassi di comando ai corrispondenti dei nuovi prodotti.

● **Felicissimo possessore di un Commodore 64 dotato di Drive e stampante MPS 802 (ottima tra l'altro) uso prevalentemente come software il Calc Result, il quale mi consente di aggiornare e archiviare con efficienza e puntualità la mia contabilità. Dopo aver passato ore (per non dire giorni) nello studio delle innumerevoli possibilità da quest'ultimo offerte, mi sono arenato comprensibilmente, in quanto non esiste spiegazione esauriente nell'apposito manuale, davanti all'istruzione DIF-FILE. Come potrete verificare, tutto è rimandato alle cortesie del «Data Interchange Format Clearinghouse» di Cambridge. Gradirei che voi ampliaste le conoscenze a questo riguardo, in caso contrario mi vedrò costretto a rinunciare alla favolosa possibilità d'intersecare i «data» tra software.** *(Adalberto Berselli - Bologna).*

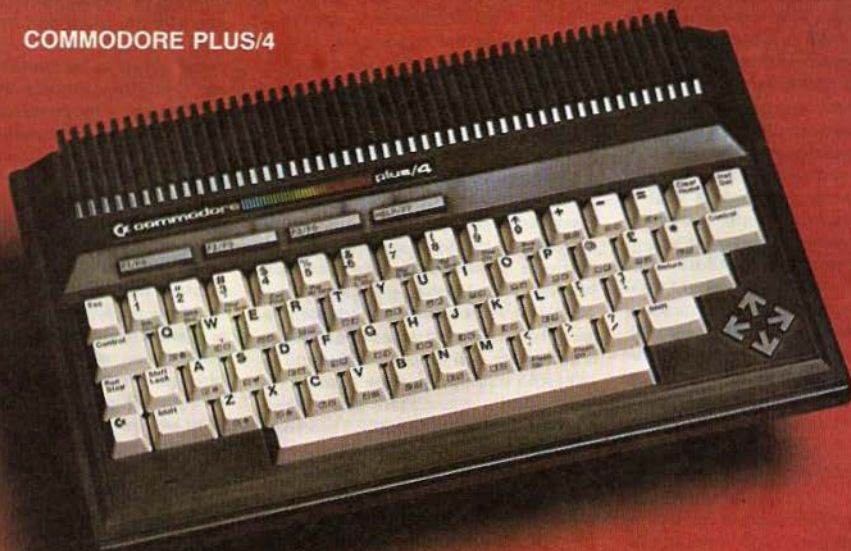
□ Il sistema che utilizza i DIF-FILE tende a fornire una intercambiabilità dei dati. Questo fatto tende a creare uno standard tale che altri pacchetti software possono attingere a dati elaborati con altri package. Così dei dati creati con il Calc Result possono essere utilizzati con ad esempio un word processor.

● **Vorrei sapere se esistono i manuali di istruzioni in italiano della stampante MPS-801 e del floppy disk 1541. E' possibile sapere come fare per accedere ai dati su disco non in modo sequenziale ma sul C64?**

□ La Commodore italiana ha iniziato ad inserire nelle confezioni del floppy disk 1541 anche un libretto in italiano. Se nella sua scatola ha trovato quello in inglese non si disperi, senza alcun dubbio il suo rivenditore sarà certo disponibile per farle delle fotocopie (a pagamento o no, dipende!). Invece per la stampante MPS-801 esiste solo quello originale.

● **Ho avuto modo di leggere il n. 2 di Commodore ed ho trovato molto interessante gli articoli della rivista. Possiede un C64. Vi sarei particolarmente grato se nei prossimi numeri pubblicaste qualche programma di utilità sui seguenti argomenti: gestione di condomini - contabilità semplificata - stampa di lettere, atti, ecc. - sviluppo di sistemi per il gioco del totocalcio. Chiedo inoltre a proposito dell'articolo comparso sul n. 2 (Stampa Fatture), se è possibile sostituire con un opportuno comando di allargamento caratteri, tutta la parte grafica inerente la stampa dell'intestazione che dia come risultato lo stesso da voi ottenuto.** *(Rossi Ferruccio).*

□ Argomenti inerenti a programmi (o meglio a procedure gestionali) richiedono indubbiamente molto spazio sulla rivista, ciò nonostante quando si potrà trovare una relazione valida, questa troverà le pagine necessarie. Il programma stampa fatture può prevedere l'allargamento dei caratteri in stampa, poichè ogni stampante ha una serie comandi atti a far eseguire l'allargamento che l'attivazione grafica, NLQ, ecc. Ecco che si rende necessaria la consultazione del relativo manuale al fine di conoscere quali e quanti CHR$ dovranno essere inviati. Nell'articolo «Stampa Fattura» vengono citati quei codici che fanno eseguire alla MPS-801 determinate funzioni (vedi righe 333 fino alla 335).

COMMODORE PLUS/4

ESCLUSIVO

Da questo numero di Commodore tutti i listati riportano in testa le informazioni relative ai tipi di computers sui quali è possibile utilizzare il programma in oggetto.

Noi per primi, quindi, forniamo listati compatibili anche con i nuovi prodotti Commodore: il C16 e il Plus 4.

COMMODORE 16

● Ho conseguito nel 1983 la Maturità Tecnica Industriale in Informatica a Catania, e dal febbraio 1984 ho trovato impiego presso un concessionario Commodore della provincia come collaboratore tecnico software/hardware. Mi occupo in particolare della riparazione delle macchine guaste. Ma in tal senso ho potuto constatare che manca una documentazione approfondita riguardo ai componenti usati ed ai difetti rilevabili quando qualcuno di essi si guasta, con conseguente eccessiva perdita di tempo nella ricerca della causa del malfunzionamento. Allora i miei quesiti sono i seguenti: esistono pubblicazioni che trattano specificamente problemi tecnici? E se sì, potreste indicarne qualcuna? Tutto questo ovviamente per una migliore qualificazione professionale, per chi, come me, opera nel settore debuging, con conseguente vantaggio per l'utente finale. *(Lucio Toscano - Catania).*

● Sono un non felice possessore di un computer Commodore 64. Ho scelto tale computer per le alte prestazioni che tale macchina offriva, per l'affidabilità e, non di secondaria importanza, per il prezzo sicuramente concorrenziale ad altri computer analoghi. Però più vado avanti e più mi accorgo di non aver fatto una buona scelta. Innanzitutto all'atto dell'acquisto mi vedo rilasciare dal negoziante una garanzia di tre mesi. Non sono riuscito ad accertare se la garanzia di tre mesi è quella rilasciata proprio dalla Commodore oppure di una garanzia del rivenditore. In ogni caso la cosa mi avrebbe lasciato indifferente (entro certi limiti), se il computer avesse funzionato a dovere. Di fatto, dopo pochi giorni dall'acquisto, mi accorgo che il C64 tende molto spesso a bloccarsi inspiegabilmente, mandando - e scusate il termine - a quel paese programmi che delle volte avevano richiesto ore di lavoro al computer. Il fenomeno si manifesta dapprima cambiando lettere e numeri del listato con simboli non introdotti precedentemente; poi alla richiesta del run o del listato, il computer o si blocca eliminando totalmente il controllo della tastiera oppure inizia a visualizzare una interminabile serie di SYNTAX ERROR, ad ogni ulteriore richiesta di run o di list. A nulla vale ovviamente premere simultaneamente i tasti run/stop e restore. Il computer torna a «funzionare» solo spegnendolo e riaccendendolo. Devo sottolineare che sono sicuro di non commettere nessun errore durante la stesura del programma, che non introduco parole chiave del basic tipo «wait» o simili o valori che possono far bloccare la memoria del computer anche se delle volte questi compaiono come passo di programma. Ho lavorato per anni con un HP 85 (computer di tutt'altro mondo) che non ha mai manifestato fenomeni di questo tipo. Ora non credo che il basso costo del C64 sia motivo di così bassa affidabilità. Durante il periodo di garanzia ho portato il computer dal rivenditore, il quale oltre a non aver capito bene il problema (da notare che si tratta di un negozio che va per la maggiore a Roma e che vende esclusivamente computers), mi assicurava che lo avrebbe sottoposto a un testo che avrebbe permesso di appurare la zona di origine di tale fenomeno. Alla fine, dopo due giorni, mi viene riconsegnato il C64 dicendo che quest'ultimo funziona perfettamente e che risponde al test e che per loro il computer non ha nulla (forse allora le ore perse erano solo un brutto sogno). Ho provato a spiegare che se il computer si trova, al momento del test, a funzionare perfettamente, non può presentare al test nessun difetto. Alla richiesta, forse da parte mia un po' pretenziosa, di usarlo fino a quando non si manifestava il fenomeno, mi hanno risposto che non potevano perdere tempo con il mio computer (forse perchè l'ho pagato solo 625.000 lire). Dopo questo lungo sfogo chiedo a voi che ritengo persone di fiducia, vista la vostra rivista, cosa debbo fare: debbo forse gettare il computer e quindi le 625.000 lire dalla finestra? Spero che voi possiate consigliarmi sull'origine del fenomeno, se dipende da un guasto o da un mio possibile errore od eventualmente dirmi dove mi posso rivolgere a Roma per far riavere al mio computer la capacità di non far svanire ore ed ore di paziente programmazione. Mi scuso con voi se questa lettera vi farà perdere del tempo e vi ringrazio fin d'ora per la risposta che attendo con impazienza. *(Maurizio Galdieri).*

☐ Effettivamente la garanzia per gli apparati Commodore è della durata di tre mesi, concessi dai rivenditori locali in accordo con la Commodore Italiana. Il difetto riscontrato nel suo C64 è ben difficile da focalizzare, in quanto non si sa se dipende da una zona di memoria RAM oppure ROM (con più probabilità da quest'ultima). Per ciò che concerne la documentazione tecnica, questa è possibile solo tramite i centri di assistenza dai rivenditori regionali, quindi le copie di queste documentazioni possono essere considerate confidenziali. La sintomatologia dei guasti non è in pratica disponibile se non «chiacchierando» con un tecnico compotente.

Una sola riga

Tutti i lettori che invieranno programmi costituiti da una sola riga, come ad esempio:

```
IL MOSCHINO PAZZO

1 A$="[UP][LEFT][DOWN][RIGHT]":PRI
NT"[LEFT][RVS] "MID$(A$,RND(1)*4
+1,1)"[LEFT][RVOFF]*";:FOR I=1 T
O 30:NEXT:GOTO 1
```

che verranno pubblicati sulla rivista saranno ricompensati con un libro a scelta tra: Programmo in Basic di Clizio Merli; *64 programmi per il Commodore 64* di Gloriano Rossi; *Utility e routine per il Commodore 64* di Gloriano Rossi.

I programmi potranno avere uno scopo oppure no. Ciò non importa! I lavori dovranno pervenire su carta con una piccola descrizione, citando il nome, l'indirizzo e il libro scelto.

Tutti i programmi saranno presi in dovuta considerazione.

CONVERSIONI DALL'ITALIANO ALL'INGLESE

(Ovvero: dalle unità di misura anglosassoni al sistema decimale)

di **Francesco Gatti**

Se si domanda a qualcuno dei nostri amici, beh... anche a noi stessi, cosa egli sappia riguardo al sistema di misura usato nei paesi anglosassoni, vi risponderà vagamente con dei vocaboli. Ad esempio: yarde, piedi e pollici. Sarà difficile comunque che vi dia un rapporto di conversione preciso. In un paese come il nostro, dove il sistema metrico decimale è il sistema di misura adottato ufficialmente, ciò non comporta particolari disagi.

Se però si viaggia spesso nei paesi inglesi, comprese le Americhe, o se si è in contatto col mondo degli affari, è indispensabile conoscere il sistema anglosassone. Questo perchè, nonostante il sistema metrico decimale sia il più usato nel mondo, talune misurazioni rimangono in rapporto con il sistema anglosassone.

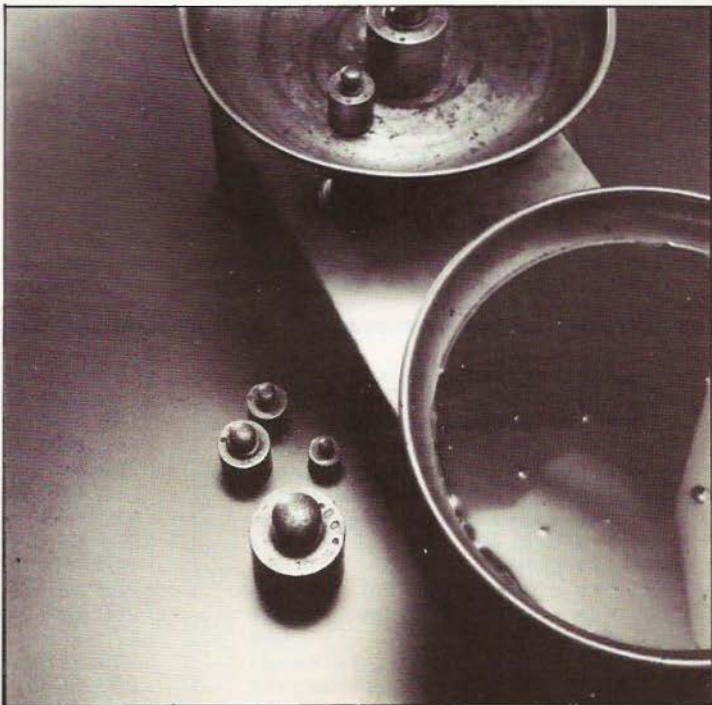
Per fare degli esempi più semplici pensate a come noi valutiamo le dimensioni di un televisore: in pollici (fra l'altro non tutti sanno che la misura dei pollici di un televisore intende proprio la diagonale dello schermo).

Sappiamo che un TV da "ventun pollici", è più piccolo di quello da "ventisei pollici", ma pochi di noi sanno dargli una misura metrica precisa e cioè, rispettivamente, 53.1 e 65.7 centimetri di diagonale. Un altro esempio: se si leggono le quotazioni di borsa si sarà notato che le sementi in genere vengono trattate in 'bushel' (pron.: buscel), e sapere che corrispondono a 36.36 litri c'è senz'altro utile.

Da qui il programma che oltre ad aprirvi il mondo delle unità di misura inglesi (non tutte, però, sta a Voi implementarlo) vi sarà utile per imparare un po' l'uso delle istruzioni READ... DATA.

Descrizione del programma

Come prima cosa ho diviso le unità di misura, ciò per facilitare la loro trattazione in:

- unità di misura di capacità;
- unità di misura di peso;
- unità di misura lineare.

Questo ha comportato l'adozione di due matrici bidimensionali che trattano una le misure inglesi, l'altra quelle metriche.

Così un solo parametro delle matrici individuerà a che gruppo appartiene l'unità: se 1 sarà di capacità, 2 di peso, 3 lineare. Il secondo parametro sarà semplicemente usato per l'ordine di successione.

Lo stesso ragionamento è stato usato nelle due matrici numeriche X e W nelle quali è inserito il valore di conversione. La parte di programma che fa tutto ciò è compresa fra le istruzioni che vanno dalla riga 200 alla 310. Nei due LOOP delle righe sopraddette avviene questo: il primo FOR di ogni LOOP determina il parametro discriminante, il secondo quello d'ordine. All'interno di questi due sta un READ che legge dalle righe di DATA poste in coda al programma e immagazzina il valore in una variabile: questa variabile viene poi uguagliata alla matrice bidimensionale. Semplice no?

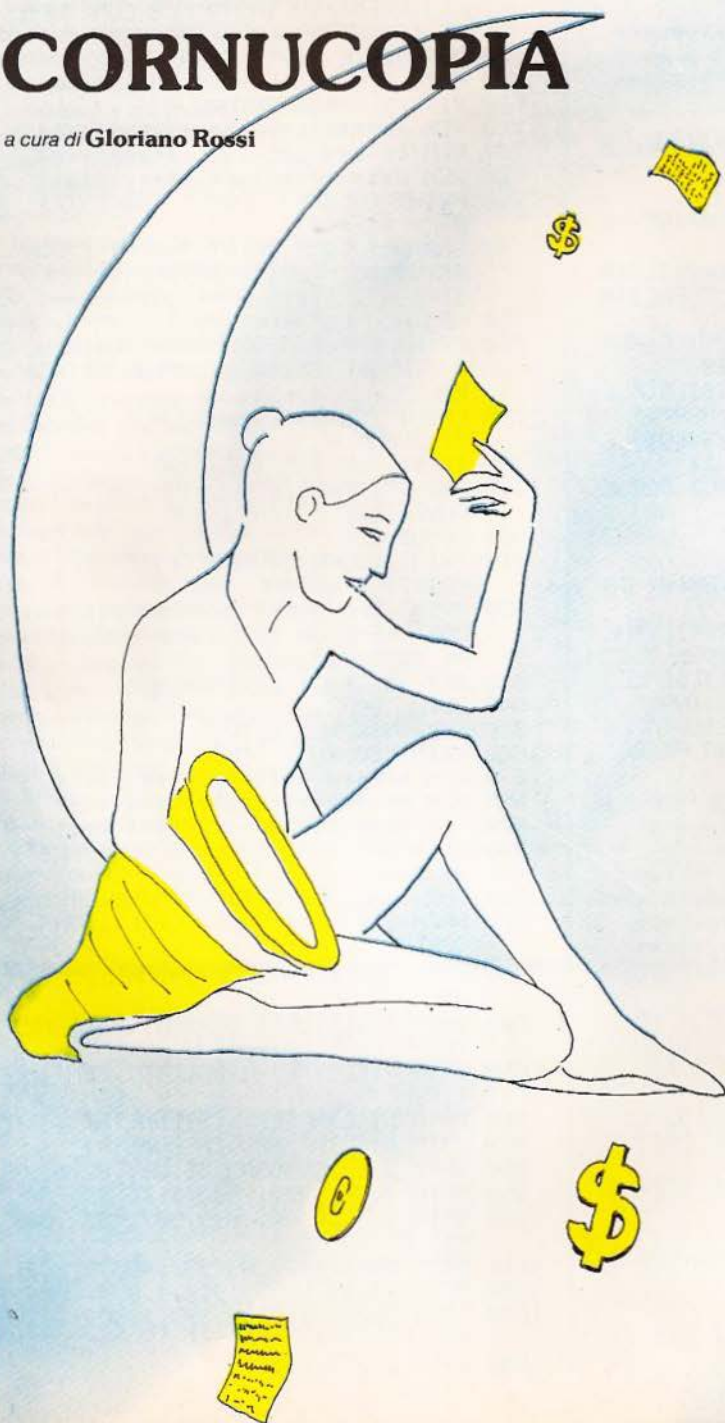
Tutto ciò permette di semplificare notevolmente la mole di calcoli e la trattazione dei vari menù: infatti basta richiamare con due semplici parametri l'unità e il suo rapporto di conversione. Il primo menù che apparirà dopo aver dato il RUN si occupa di discernere il campo sul quale si svolgerà il programma, usando per questo la variabile S, primo parametro delle matrici. Se si volesse aumentare la capacità di conversione, sarà necessario dimensionare opportunamente le quattro matrici, e quindi, si dovranno aggiornare altrettanto opportunamente i valori presenti nei DATA. ▪

```
C R O S S   R E F E R E N C E

PROGRAMMA : ARTCONVERS. PRG

-------*------------------------------ - - -
VAR.   ! LINEA DEL PROGRAMMA
-------*------------------------------ - - -
       !
A      !   580   590   740   750
A$     !   220
C$     !   230
G      !   210   220   230   240
IN$(   !   220   580   740   810
IT$(   !   230   580   650   740
K      !   270   280   290   300
Q      !   290
R      !   510   520   530   610   620   650
       !   690   770   780   810   850
RI     !   650   690   810   850
S      !   440   450   460   570   580   650
       !   690   730   740   810   850
S$     !   670   830
T      !   200   220   230   250
TE$(   !   360   370   380   390   470   570
       !   730
UN     !   630   690   790   850
V      !   260   280   290   310
W(     !   290   850
X(     !   280   690
Z      !   280
```

```
100 REM **************************
110 REM *** CONVERSIONI **********
120 REM **************************
130 REM *** GATTI FRANCESCO ******
140 REM **************************
150 REM *** SETTEMBRE 84 *********
160 REM **************************
161 REM *** VIC 20            SI ***
162 REM *** COMMODORE 64      SI ***
163 REM *** COMMODORE 4000    SI ***
164 REM *** COMMODORE 8000    SI ***
165 REM *** COMMODORE 16      SI ***
166 REM *** *********************
167 REM **************************
168 REM **************************
170 REM *** INIZIALIZZAZIONE *****
180 REM ***      MATRICI     *****
190 REM **************************
200 FOR T=1 TO 3
210 FOR G=1 TO 5
220 READ A$:IN$(T,G)=A$
230 READ C$:IT$(T,G)=C$
240 NEXTG
250 NEXTT
260 FOR V=1 TO 3
270 FOR K=1 TO 5
280 READ Z:X(V,K)=Z
290 READ Q:W(V,K)=Q
```

```
300 NEXTK
310 NEXTV
320 REM **************************
330 REM **** INIZIO PROGRAMMA ****
340 REM **************************
350 POKE 53280,2:POKE 53281,2
360 TE$(0)="CONVERSIONI"
370 TE$(1)="CAPACITA´":TE$(2)="PESO
    "
380 TE$(3)="LINEARE"
390 PRINT"[CLEAR][BIANCO]":PRINT TA
    B(12)TE$(0)
400 PRINT TAB(13)"[2 DOWN][RVS][GIA
    LLO] 1 [RVOFF][BIANCO] CAPACITA
    ´"
410 PRINT TAB(13)"[DOWN][RVS][GIALL
    O] 2 [RVOFF][BIANCO] PESO"
420 PRINT TAB(13)"[DOWN][RVS][GIALL
    O] 3 [RVOFF][BIANCO] LINEARE"
430 PRINT TAB(13)"[DOWN][RVS][GIALL
    O] 4 [RVOFF][BIANCO] QUIT"
440 PRINT TAB(13)"[DOWN]QUALE SCEGL
    I";:INPUT S
450 IF S<1 OR S>4 THEN 390
460 IF S=4 THEN 860
470 PRINT"[CLEAR][BIANCO]":PRINT TA
    B(12)TE$(0)
480 PRINT TAB(8)"[2 DOWN][RVS][GIAL
    LO] 1 [RVOFF][BIANCO] INGL-DEC"
490 PRINT TAB(8)"[DOWN][RVS][GIALLO
    ] 2 [RVOFF][BIANCO] DEC-INGL"
500 PRINT TAB(8)"[DOWN][RVS][GIALLO
    ] 3 [RVOFF][BIANCO] MENU PRINCI
    PALE"
510 PRINT TAB(8)"[DOWN]QUALE SCEGLI
    ";:INPUT R
520 IF R<1 OR R>3 THEN 470
530 ON RGOTO 570,730,390
540 REM **************************
550 REM ****** INGL-DEC **********
560 REM **************************
570 PRINT"[CLEAR]":PRINT TAB(7)TE$(
    0)":  "TE$(S)
580 FOR A=1 TO 5:PRINT TAB(8)"[DOWN
    ][RVS][GIALLO]"A"[RVOFF][BIANCO
    ][RIGHT]"IN$(S,A)"-"IT$(S,A)
590 NEXTA
600 PRINT TAB(8)"[DOWN][RVS][GIALLO
    ] 6[RVOFF][BIANCO]  MENU PRINCI
    PALE"
610 PRINT TAB(8)"[DOWN]QUALE SCEGLI
    ";:INPUT R
620 IF R<1 OR R>5 THEN 390
630 PRINT"[DOWN][GIALLO]UNITA´ DA C
    ONVERTIRE:";:INPUT UN
640 GOSUB 690
650 PRINT"[DOWN]CONVERTITO E´:"RI;:
    PRINTIT$(S,R)
660 PRINT TAB(6)"[3 DOWN][RVS][BIAN
    CO] PREMI UN TASTO PER CONTINUA
    RE [RVOFF]"
670 GET S$:IF S$="" THEN 670
680 GOTO 570
690 RI=X(S,R)*UN:RETURN
700 REM **************************
710 REM ****** DEC-INGL **********
720 REM **************************
730 PRINT"[CLEAR]":PRINT TAB(7)TE$(
    0)":  "TE$(S)
740 FOR A=1 TO 5:PRINT TAB(8)"[DOWN
    ][RVS][GIALLO]"A"[RVOFF][BIANCO
    ][RIGHT]"IT$(S,A)"-"IN$(S,A)
750 NEXTA
760 PRINT TAB(8)"[DOWN][RVS][GIALLO
    ] 6[RVOFF][BIANCO]  MENU PRINCI
    PALE"
770 PRINT TAB(8)"[DOWN]QUALE SCEGLI
    ";:INPUT R
780 IF R<1 OR R>5 THEN 390
790 PRINT"[DOWN][GIALLO]UNITA´ DA C
    ONVERTIRE:";:INPUT UN
800 GOSUB 850
810 PRINT"[DOWN]CONVERTITO E´:"RI;:
    PRINTIN$(S,R)
820 PRINT TAB(6)"[3 DOWN][RVS][BIAN
    CO] PREMI UN TASTO PER CONTINUA
    RE [RVOFF]"
830 GET S$:IF S$="" THEN 830
840 GOTO 730
850 RI=W(S,R)*UN:RETURN
860 PRINT"[CLEAR]":END
870 REM **************************
880 REM **************************
890 REM **************************
900 DATA QUARTERS,LITRI,BUSHELS,LIT
    RI
910 DATA GALLONI,LITRI,PINTE,LITRI
920 DATA O,O,QTLI INGL.,CHILI,TONN.
    CORTA
930 DATA CHILI,TONN.LUNGA,CHILI,LIB
    BRE
940 DATA CHILI,ONCE,GRAMMI,YARDE,ME
    TRI
950 DATA PIEDI,METRI,POLLICI,CENTIM
    ETRI
960 DATA MIGLIA TER.,CHILOMETRI
970 DATA MIGLIA MAR.,CHILOMETRI
980 DATA 291.2,.00343,36.36,.0275
990 DATA 4.54,.220,.568,1.76
1000 DATA 1,1,50.80,.019,907.185,.00
     11
1010 DATA 1016.047,.00098,.453,2.207
1020 DATA 28.35,.035,.914,1.094,.304
1030 DATA 3.289,2.539,.393,1.609,.62
     1
1040 DATA 1.853,.539
```

CORNUCOPIA

a cura di **Gloriano Rossi**

Quando una idea non è completamente tua, è bene dirlo per primi! E' proprio il caso di questa nuova rubrica che vede il suo nascere in questo numero di Commodore.

L'idea di realizzare una serie di questo genere è nata sfogliando le riviste americane, le quali periodicamente dedicano qualche pagina a varie spigolature su piccoli problemi, gadget, etc.

Il nome Cornucopia mi è venuto in mente pensando che i contenuti di questa rubrica possano essere dei veri e propri doni. Infatti la Cornucopia era quello strano corno dal quale le dea Fortuna traeva doni che alla cieca proferiva agli uomini.

Ecco qui, allora, l'edizione italiana che vuole esibire il «merge» delle varie idee: valide, americane più idee nostre, ancora più valide.

Quale è e quale sarà sempre il contenuto di Cornucopia?

Tutti gli argomenti che non giustificano un articolo vero e proprio, tutte quelle spigolature che per propria natura possono occupare una piccola parte della rivista, trovano posto in questa rubrica. Come vedrete, tutti i «doni» sono e saranno numerati con il sistema esadecimale e quindi incominceremo con il $01, poi il $02, e così via fino ad arrivare... a $FFFF.

Chi scrive i vari $nn? Io, tu, lei professore, dottore, ingegnere, beh... chi ha una idea scriva. Ogni argomento è valido. Non pensate che dire A+B=C sia banale. A qualche lettore potrà interessare.

Ogni $nn sarà sempre firmato (è chiaro che i primi li ho fatti io i 2 KH Gloriano Rossi), a meno che l'autore non desideri il contrario. E... i migliori $nn saranno ricompensati adeguatamente.

Inviate i vostri $nn a:
Spett. Rivista COMMODORE
rubrica Cornucopia
Gloriano Rossi

Viale Famagosta, 75
20142 Milano

$ 01

Tempo di attesa. Il sistema più preciso per far passare il tempo nel vostro computer è quello di utilizzare la variabile di sistema TI$. Spesso si utilizza questo sistema:

```
FOR I=1 TO 1000: NEXT
```

Questo loop crea un ritardo di circa un secondo. Invece:

```
100 T=TI: PAUSA=60
110 IF TI< T+PAUSA THEN 110
```

ci procura esattamente una pausa di 1 secondo.

Ciò è dovuto al fatto che si sfrutta il CLOCK del computer che non può certo fallire.

Con PAUSA=30 si ottiene una attesa di mezzo secondo.

Con PAUSA=90 si ottiene una attesa di un secondo e mezzo.

Con PAUSA=120 si ottiene una attesa di due secondi, e così via.

$ 02

Repeat sui tasti. Sia per il Vic 20 che per il Commodore 64, per ottenere la variabilità di repeat sui tasti, si deve eseguire:

```
POKE 650,128
```

e tutti i tasti della tastiera sono abilitati al repeat automatico.

```
POKE 650,127
```

e nessun tasto della tastiera è abilitato al repeat automatico.

```
POKE 650,0
```

con questo comando si riporta l'operabilità della tastiera alle normali condizioni, proprio come quando si accende il computer.

Ovvero il repeat automatico è presente solo per i tasti: Spazio, Cursore a destra e sinistra, Cursore in sù ed in giù, Insert e Delete.

$ 03

System reset. Per poter simulare da programma o tramite la tastiera il system rest, ovvero quella azione che equivale allo spegnimento ed alla riaccensione del computer, dovete eseguire:

Per il Vic 20

```
SYS (64802)
```

Per il Commodore 64

```
SYS (64738)
```

Per i Commodore della serie 4000, 8000 e 8096

```
SYS (64790).
```

$ 04

Cancella numero di riga. Provate ad eseguire le istruzioni seguenti:

```
0 REM "" [DEL] [RVS]TTTTTT [RVOFF] QUESTA E' UNA INTESTAZIONE
```

ed ora eseguité il comando di LIST. Visto che bello!!??

$ 05

I tasti funzione. Come si possono utilizzare i tasti funzione direttamente in BASIC?

Sembra che l'unico sistema sia quello del CHR$. Sappiamo infatti che i tasti funzione hanno un diretto riferimento numerico nella tabella ASCII.

Premendo il tasti F1 si ha il codice 133.
Premendo il tasto F3 si ha il codice 134.
Premendo il tasto F5 si ha il codice 135.
Premendo il tasto F7 si ha il codice 136.

E se i medesimi tasti funzione sono premuti contemporaneamente al tasto SHIFT:

Premendo il tasto F2 si ha il codice 137.
Premendo il tasto F4 si ha il codice 138.
Premendo il tasto F6 si ha il codice 139.
Premendo il tasto F8 si ha il codice 140.

Verifichiamo quanto detto con questo semplice programmino, che vuole essere un suggerimento e non l'unica soluzione.

Dopo di che è possibile eseguire a seconda delle necessità:

```
300 ON XI GOTO ..............................................
```

oppure

```
310 ON XI GOSUB ..............................................
```

Dove i puntini devono essere sostituiti da relativi numeri di riga.

$ 06

Syntax error nel LIST. E' possibile forzare una condizione di errore, per nulla dannosa ai fini dell'esecuzione del programma, ma tale che in seguito all'istruzione LIST venga interrotto il listato con questo semplice e nevrotico messaggio:

? SYNTAX ERROR

Come fare? Sulla riga precedente a quella dalla quale deve essere interrotto il listato si deve aggiungere semplicemente la seguente istruzione:

: REM [SHIFT L]

Si, proprio il tasto shift più il tasto L provocano questa divertente interruzione del listato.

$ 07

Le insidie del Basic. Occorre fare buona attenzione alle variabili da utilizzare ed alle istruzioni BASIC. Infatti una variabile tipo F può creare degli errori con l'istruzione BASIC OR.

Cosa fare allora? I casi sono due: o cambiare il nome della variabile, oppure separare, la dove può esistere un errore a torto, le istruzioni BASIC dalle relative variabili insidiose.

Così:

IFA=FORA=NTHEN35

è meglio che venga scritto:

IF A=F OR A=N THEN 35

Infatti nel primo caso l'interprete BASIC si sarebbe confuso ed avrebbe inteso:

IF A= FOR A=N THEN 35

E ancora:

FORI=STOP: NEXT

è meglio che venga scritto:

FOR I=S TO P: NEXT

E' implicito nel primo caso la variabile S e la variabile P risultano incongruenti con l'istruzione TO. Risultato erraro: uno STOP non interpretabile.

$ 08

Maiuscolo/Minuscolo. Premendo contemporaneamente il tasto di SHIFT ed il tasto con il simbolo della Commodore (in basso a sinistra) si ottiene il passaggio da minuscolo in maiuscolo e viceversa.

A volte però può essere dannoso al fine estetico dell'esecuzione del programma.

Si rende quindi necessaria la possibilità di abilitazione e disabilitazione di questa funzione diretta da tastiera.

Per disabilitare si esegue:

PRINT CHR$(8) oppure

PRINT "[CTRL H]

per riabilitare si esegue:

PRINT CHR$(9) oppure

PRINT "[CTRL I]

Il medesimo effetto si ottiene con:

POKE 657,128 per disabilitare e

POKE 657,0 per riabilitare

$ 09

Disabilitazione del LIST. Eseguendo il comando:

POKE 775,200

si ottiene la disabilitazione del comando LIST.

Dopo aver eseguito questa POKE ed impartendo ugualmente il comando BASIC diretto LIST si ottiene esclusivamente la pulitura dello schermo e... nulla più.

Per ritornare alla normale funzione, riabilitare cioè il LIST, si deve semplicemente eseguire:

POKE 775,167

Per i possessori del Vic 20:

POKE 775,0 disabilita

POKE 775,199 riabilita

$ 0A

Disabilitazione della tastiera. Sia per il Vic 20 che per il Commodore 64, è possibile rendere inattiva la tastiera. Questo fatto può essere utile al fine di evitare che in fase di elaborazione venga premuto qualche tasto a torto.

Con il comando:

POKE 649,0

si ottiene la disabilitazione della tastiera, mentre con:

POKE 649,10

si ha la riabilitazione.

Sarà quindi possibile attivare la tastiera solamente quando sia necessaria un input di un dato o deve essere premuto un tasto qualsiasi per proseguire

l'elaborazione.

Attenzione: la disabilitazione non influisce sui comandi STOP e STOP/RESTORE, i quali dovranno essere disabilitati con altri opportuni comandi.

$ 0B

STOP no STOP. Nel $ 0A abbiamo visto come sia possibile per il Vic 20 e per il Commodore 64 rendere inattiva la tastiera. Eseguendo però:

POKE 808,239 (per il Vic 20 POKE 808,114)

si disabilita il solo tasto di STOP, mentre l'Azione di STOP/RESTORE rimane inalterata.

Con:

POKE 808,225 (per il Vic 20 POKE 808,100 oppure POKE 808,127)

si disabilita sia lo STOP che la funzione STOP/RESTORE e non solo: anche la funzione del comando LIST. Infine per riportare tutto alla normalità si deve eseguire:

POKE 808,237 (per il Vic 20 POKE 808,112).

$ 0C

La riga fantasma. A volte, nei programmi, può essere piacevole vedere separate le varie routines o i vari blocchi di programma. Come si fa? I casi sono due: si utilizzano righe con REM, oppure si possono creare delle righe fantasma, cioè delle righe senza alcun contenuto che non influiscono in alcun modo sull'esecuzione del programma se non quello di aumentare di un valore infinitesimale di tempo l'esecuzione stessa.

Come si crea una riga fantasma? Così:

100 [SHIFT SPAZIO] [SPAZIO] [SHIFT SPAZIO]

e quindi

RETURN

Semplice no?

$ 0D

SHIFT-COMMODORE-CONTROL. Potrebbe essere utile sapere, sempre da programma, se uno o più tasti di controllo siano o no premuti. Esiste per questo scopo una locazione ben precisa di memoria (in pagina 3) che dice al sistema operativo se si è premuto il tasto SHIFT, oppure il tasto COMMODORE, o il tasto di CONTROL oppure due o tre di questi contemporaneamente.

Ebbene questa locazione, comune per il Vic 20 e per il Commodore 64, è accessibile anche in BASIC. Il valore normale, quando cioè nessuno dei tre tasti in questione è premuto, equivale a 0.

Questo zero cambia quando è premuto:

nessun tasto: 0
SHIFT: 1
COMMODORE: 2
CONTROL: 4
SHIFT-COMMODORE: 3
SHIFT-CONTROL: 5
COMMODORE-CONTROL: 6
SHIFT-COMMODORE-CONTROL: 7

Notate come i risultati siano binariamente espressi?

nessun tasto	00000000	= 0
SHIFT	00000001	= 1
COMMODORE	00000010	= 2
CONTROL	00000100	= 4
SHIFT-COMMODORE	00000011	2+1 = 3
SHIFT-CONTROL	00000101	= 4+1 = 5
COMMODORE-CONTROL	00000110	= 4+2 = 6
SHIFT-COMMODORE-CONTROL	00000111	= 4+2+1 = 7

Per poter verificare quanto appena detto provate a eseguire questo semplice programma di una sola riga BASIC.

```
10 PRINT "[CLEAR]" PEEK(653): GOTO 10
```

Dopo il RUN provate a tenere premuto uno o più di quei tasti tema di questa spigolatura.

$ 0E

Fissativo. Può risultare utile a volte rendere difficile l'azione di cancellare un programma su disco. Ciò per evitare errate manovre o inconvenienti dannosi. Per fare questo, è sufficiente far precedere il nome del file da uno spazio shiftato.

Il risultato che si ottiene si può osservare chiedendo la directory del disco:

LOAD "$",8 e poi LIST

Vedremo che il programma che abbiamo chiamato

NOME apparirà in questa maniera:

""NOME PRG

E' chiaro che se si volesse cancellare questo file-programma dichiarando ""NOME, si avrà un messaggio di errore (ILLEGAL QUANTITY ERROR). Se invece provassimo con: "NOME", si avrebbe un bel: FILE NOT FUND.

Come si fa a cancellare o semplicemente richiamare un programma salvato con questo artificio? Semplicemente facendo precedere il nome del programma dallo spazio shiftato, così come lo abbiamo salvato, oppure si inserisce al posto dello spazio shiftato un punto di domanda. Così ad esempio, se volessimo caricare in memoria, si deve eseguire:

```
LOAD "[SHIFT SPAZIO] NOME",8
```

oppure

```
LOAD "?NOME",8
```

$ 0F

Un difetto del DOS. Mentre si digita o si edita un programma, è ottimo costume eseguire per sicurezza e periodicamente un SAVE. I manuali dei floppy drive tipo 1541, 2031, 4040, 8050 etc. ci consigliano di eseguire il comando in questa maniera:

```
SAVE "@0:NOME",8
```

se il programma è già presente sul dischetto. Questa azione infatti dice al DOS (Disk Operating System) che se il medesimo nome risulta già presente sul disco, questo va sostituito da quello che sta per arrivare dalla memoria centrale.

Sfortunatamente a volte questo sistema di SAVE può creare dei danni sia al programma in salvataggio che ad altri file di dati o di programmi. Un difetto di tutti i DOS Commodore.

La migliore soluzione per ovviare a questo inconveniente è quella di cancellare il programma su disco e quindi eseguire il normale SAVE. Oppure rinominare il vecchio programma, poi salvare il nuovo e quindi cancellare il vecchio rinominato.

Vediamo in pratica come fare.

```
OPEN 15,8,15, "S0:NOME":CLOSE 15
SAVE "0:NOME",8
```

oppure:

```
OPEN 15,8,15,"R0:NOMEOLD=NOME": CLOSE 15
SAVE "0:NOME",8
OPEN 15,8,15, "S0:NOMEOLD": CLOSE 15
```

Con il BASIC 4.0 si può semplificare:

```
SCRATCH D0,"NOME"
DSAVE D0, "NOME"
```

oppure

```
RENAME D0, "NOME" TO "NOMEOLD"
DSAVE D0, "NOME"
SCRATCH D0, "NOMEOLD"
```

Ma un sistema più semplice per rendere veloce questa azione periodica può essere quella di inserire queste istruzioni direttamente nel programma, proprio alla fine:

```
63500 X$="NOME"
63510 OPEN 15,8,15
63520 PRINT≠15,"S0:"X$"OLD"
63530 PRINT≠15,"R0:"X$"OLD="X$
63540 CLOSE 15
63550 SAVE "0:"+X$,8
```

Eseguire periodicamente, quando cioè ci viene in mente:

```
RUN 63500 oppure
GOTO 63500
```

$ 10

Dopo la REM. I trucchetti che si possono attuare in un listato sono veramente molti! Eccone uno alquanto simpatico. Digitando in un primo momento:

100 REM "PROVA e quindi RETURN

In un secondo momento si torna con il cursore proprio dopo i doppi apici e si esegue:

[rvs][3 insert][3 SHIFT M]

e quindi ancora RETURN

Fatto questo provate ad eseguire il LIST.

$ 11

Dopo la M. Ecco un altro trucchetto da inserire in un listato.

Provate a digitare:

100 REM "CODICE MASCHERATO

e quindi RETURN

Poi si torna con il cursore proprio sopra la lettera M di MASCHERATO e si trasforma detta M in negativo eseguendo semplicemente:

[rvs]M e quindi ancora RETURN

Eseguendo il LIST rimmarrà visibile solamente:

100 REM "CODICE

mentre la parola MASCHERATO, sia pure ancora esistente, non verrà visualizzata, cioè verrà "mascherata".

LO SCOMPATTATORE PER TUTTI I TIPI DI COMMODORE

di **Giancarlo de Cobelli**

Si chiama scompattatore. E' una utility che svolge esattamente la funzione contraria rispetto a quella pubblicata sul numero precedente di Commodore.

Come vedremo meglio in seguito, ci permette di separare il programma mettendo una istruzione per linea. Là dove è possibile, naturalmente. Questo fatto può risultare utilissimo per analizzare nel miglior modo un programma e permetterci così di afferrarne il perfetto funzionamento.

Qualche lettore potrà obiettare che esiste già in circolazione una versione di scompattatore, ma la versione che pubblichiamo evita alcuni errori di svolgimento. Il listato è fra l'altro implementato con la routine di input controllato e di fine programma, e non solo: ci permette anche di scompattare la linea 0.

Non capita spesso di trovare programmi che inizino alla linea 0: la logica del programmatore porta sempre ad iniziare la numerazione di un nuovo programma dalla linea 10 o 100. Abbiamo ritenuto opportuno dare anche questa possibilità perchè può sempre capitare di trovare un programma che inizi alla linea 0, come del resto a me è capitato spesso.

Funzionamento

A differenza del compattatore, lo scompattatore lavora linea per linea. Dopo aver letto il puntatore della prossima linea ed il numero di riga corrente, va cioè ad analizzare subito la linea corrente ponendo i caratteri letti in una variabile V(X). Ogni volta che nella lettura dei caratteri viene riconosciuto un token prestabilito (vedremo dopo nei dettagli quali sono questi tokens) il controllo viene passato alla routine di scompattamento.

Analizziamo ora linea per linea il listato:

100-116. Commento informativo sul programma di nessuna utilità pratica.

117. Pulizia delle variabili esistenti e dimensionamento della matrice unidimensionale (V(X).

118-136. Nella **riga 123** sono contenute le POKE che permettono di modificare i colori di schermo (per il Vic 20 bisogna digitare 36879 e 36880, per tutti gli altri Commodore le due POKE si devono eliminare) ed il codice ASCII 14 che trasforma i caratteri shiftati in caratteri maiuscoli. La **riga 124** contiene tra virgolette le diciture «scompattatore» in caratteri shiftati. Dalla **riga 125** alla **riga 131** ci sono le istruzioni di utilizzo del programma. Le **linee 132-136** servono per assumere i vari input richiesti dal programma e per controllare che quest'ultimi siano corretti.

137-144. Apertura del canale di controllo errori e del canale di scrittura. Salto alla routine che controlla se ci sono errori nel disco e trasformazione del nome del programma sorgente con l'aggiunta del suffisso '/S'. La **riga 142** cancella il programma, se esiste, che possiede il medesimo nome di quello che deve essere generato (questo fatto potrà capitare quando si esegue per due volte l'esecuzione sullo stesso programma sorgente). In seguito apre il canale di scrittura e ricontrolla se ci sono errori sul disco.

145-171. Nella **linea 151** vengono letti i primi due caratteri che contengono il puntatore alla prossima linea e che vengono riscritti sul disco destinazione. La variabile F viene posta uguale ad 1 poichè serve per controllare se la linea è già stata presa in considerazione per controllarla (F=1).

La variabile LN (numero riga del programma destinazione) viene posta uguale alla variabile NL (numero riga programma sorgente); controllo della variabile LK che indica, in questo caso, quando il programma è stato scompattato e stampa del numero di linea che si sta esaminando. Le **linee 158-159** scrivono sul disco destinazione due caratteri di controllo ed il numero di riga. Dopo di che c'è la routine di lettura dei caratteri con memorizzazione nell'array V(X) ed incremento dello stesso **(160-163).** Lettura del numero di riga e controllo se la riga va esaminata, tramite la variabile LK. Leggo altri due caratteri e trasformo i due caratteri che contengono l'indirizzo alto e basso del numero di linea in numero. La **linea 171** controlla il valore di F per poter passare al controllo dei tokens (F=1) o ritornare per poter stampare il numero di linea in questione (F=0).

172-192. Controllo dei caratteri contenuti nella matrice V(X) partendo da X=1. Se la lettura corrisponde ad un due punti (codice ASCII 58) salto alla routine che controlla se i caratteri che vengono dopo corrispondono ad i tokens BASIC altrimenti scrittura del ca-

rattere letto sul disco destinazione; incremento della variabile LN e controllo se maggiore di NL ed in tal caso scrittura del carattere che era stato letto.

La **linea 182** serve per chiudere una linea dopo che è stata scritta sul disco destinazione. Trasformazione del numero di linea in codice ASCII per la scrittura sul disco destinazione ed incremento della matrice V(X). Se il carattere in questione corrisponde ad uno spazio (32) od un due punti incremento della variabile X e ritorno alla lettura della matrice V(X).

La **linea 189** controlla se il carattere letto è escluso dai codici ASCII indicati che contengono tutti i tokens delle istruzioni BASIC, ed in tal caso salto alla routine di scompattamento dove si controlla che V(X) sia o non sia il codice ASCII 34 (virgolette).

La **linea 190** se trova V(X) uguale al token END (128) o maggiore del codice ASCII 153 (dopo questo si trovano tutte le parole chiave del BASIC non utilizzabili nel programma ad accezione delle tabulazioni o di THEN, STEP, TO che comunque non hanno nessuna importanza poichè le prime non influiscono sul concetto con il quale funziona lo scompattatore e le seconde sono sempre precedute da un FOR od un IF) salto alla routine di scrittura del carattere sul disco destinazione.

Le **linee 191 e 192** controllano se V(X) corrisponde ad i codici ASCII che comprendono i tokens di salto condizionato o incondizionato ed alcuni altri come IF, RESTORE, RETURN, REM e in tal caso viene eseguito il medesimo salto descritto per la **linea 189.**

193-208. Le **linee 198-199** formano una routine di scrittura sul disco destinazione condizionata solo dal fatto che se V(X) è uguale a 0, allora torna alla routine di lettura e scrittura dell'indirizzo di partenza; gli altri comandi di scrittura che troviamo in queste righe sono utilizzati esclusivamente dalle routine di controllo dei tokens. La **riga 201** controlla se il carattere letto non corrisponde alle virgolette (34) e in caso scrive il carattere; controlla se V(X) è maggiore di 0 e se è vero incrementa la variabile X e ritorna al controllo dei tokens, altrimenti non incrementa X e va alla routine di lettura e scrittura dell'indirizzo di partenza **(linee 206-208).**

```
C R O S S   R E F E R E N C E

PROGRAMMA : SCOMPATTATORE

-------*-------------------------------- - - -
VAR.   ! LINEA DEL PROGRAMMA
-------*-------------------------------- - - -
       !
C$     !   152    231    233    234
DD$    !   134    135    143
DO$    !   132    133    138    142
EM$    !   236    240
EN     !   236    237    240
ES     !   236    240
ET     !   236    240
F      !   153    171
H      !   183    184    185
K      !   217
L      !   184    185
LH     !   159    170
LK     !   156    165    168
LL     !   159    169
LN     !   155    157    180    181    183    184
ND$    !   141    142    143
NL     !   155    167    181
NP$    !   136    138    141
SN$    !   220    222
T      !   162    163    165    167    170    230
       !   233    234
T1     !   152    165    167    169    230
V(     !   117    162    178    187    189    190
       !   191    192    198    199    201    202
       !   204    206    207
X      !   160    162    163    177    178    179
       !   186    187    189    190    191    192
       !   198    199    201    202    203    204
       !   206    207
```

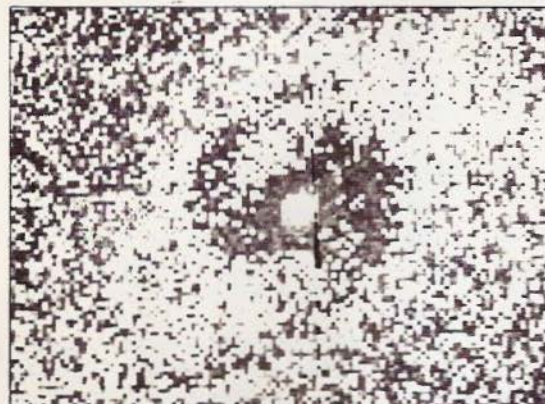
Se il carattere corrisponde a virgolette allora scrive il carattere, incrementa la variabile X e torna a scrivere un altro carattere a meno che l'ultimo carattere non corrisponda a virgolette od a 0 perchè in tal caso si andrebbe alla routine prima descritta **(linee 206-208).**

209-223. Chiusura del programma scritto sul disco destinazione tramite la scrittura di due 0 e chiusura dei canali di lettura, scrittura, controllo errori. Questo avviene quando durante la lettura dei puntatori alla prossima riga si incontrano due zeri che segnalano che il programma che si sta esaminando è finito. Avviso di fine scompattamento ed attesa

dell'input per fine lavoro o nuovo scompattamento. In caso di fine lavoro viene data la sys di azzeramento della memoria BASIC che per il Commodore 64 corrisponde a 64738 (per il Vic 20 è 64802 e per il Commodore della serie 4000 e 8000 è 64790).

224-240. Queste linee corrispondono alle subroutines di lettura di due caratteri dal disco e del controllo di errore del disco. La lettura dei caratteri dal disco avviene nelle linee **229-235** dove i caratteri letti vengono posti nelle variabili T e T1 mentre la routine di controllo di errore su disco avviene dalla **linea 236** alla **linea 240** con il solito metodo descritto anche dal manuale d'uso del 1541.

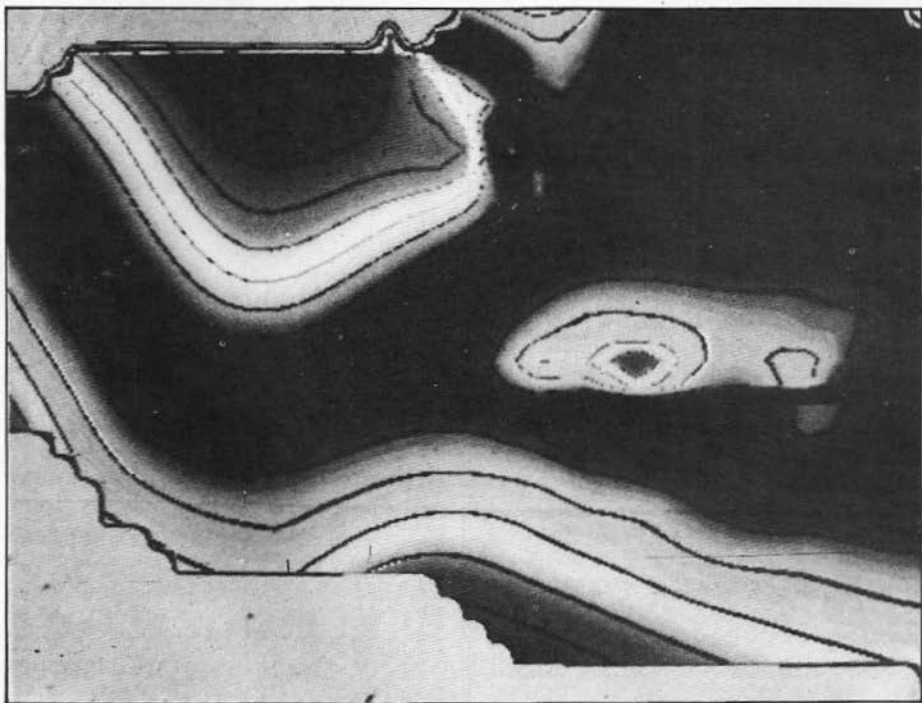
Breve conclusione

Con la speranza che queste due utility da me modificate e commentate possano essere utili per il vostro scopo, vi ricordo che per un migliore utilizzo sia del compattatore, pubblicato sullo scorso numero, che dello scompattatore, è notevolmente meglio l'esecuzione in forma compilata, naturalmente con l'austrospeed con il PetSpeed.

SCOMPATTATORE

```
100 REM *************************
101 REM *                       *
102 REM *     SCOMPATTATORE     *
103 REM *                       *
104 REM *      MODIFICA DI      *
105 REM *                       *
106 REM *       GIANCARLO       *
107 REM *                       *
108 REM *      DE  COBELLI  .   *
109 REM *                       *
110 REM *   V.LE DEI FIORI 65   *
111 REM *                       *
112 REM *  CUSANO MILANINO  MI  *
113 REM *                       *
116 REM *************************
117 CLR :DIM V(256)
118 REM *************************
119 REM *                       *
120 REM *      ISTRUZIONI       *
121 REM *                       *
122 REM *************************
123 POKE 53280,0:POKE 53281,0:PRI
    NTCHR$(14)
124 PRINT"[VERDE][CLEAR]" TAB(12)
    "[RVS] ●┤╲▔♠||♠|┌_▔ [2 DOWN]
125 PRINT"●UESTA UTILITY SCOMPATT
    A UN PROGRAMMA PO
126 PRINT"STO NEL DRIVE PRESCELTO
      DIVIDENDO TUTTE
127 PRINT"LE LINEE  POSSIBILI LAS
    CIANDO COSI' SOLO
128 PRINT"UNA ISTRUZIONE PER LINE
    A.[2 DOWN]
129 PRINT"¬L PROGRAMMA SCOMPATTAT
    TO AVRA'LO STESSO
130 PRINT"NOME, MA SEGUITO DAL SU
    FFISSO '/S' E SA-
131 PRINT"RA' SALVATO SUL DRIVE P
    RESCELTO.[DOWN]"
132 INPUT "[RVS] ¯RIVE SORGENTE
      #";DO$
133 IF DO$<CHR$(48) OR DO$>CHR$(4
    9) THEN 132
134 INPUT "[RVS] ¯RIVE DESTINAZIO
    NE#";DD$
135 IF DD$<CHR$(48) OR DD$>CHR$(4
    9) THEN 134
136 INPUT "[RVS] ╱OME PROGRAMMA
      ";NP$
137 OPEN 15,8,15
138 OPEN 5,8,5,DO$+":"+NP$+",P,R"
139 GOSUB  236
140 PRINT"[CLEAR]OK, STO LAVORAND
    O SULLE LINEE....[DOWN]"
141 ND$=LEFT$(NP$,14)+"/S"
142 PRINT#15,"S"+DO$+":"+ND$
143 OPEN 6,8,6,DD$+":"+ND$+",P,W"
144 GOSUB 236
145 REM *************************
146 REM *                       *
```

```
147 REM *  LETTURA E SCRITTURA  *
148 REM * INDIRIZZO DI PARTENZA *
151 GOSUB 229
152 PRINT#6,CHR$(T1);C$;
153 F=1
154 GOTO 164
155 LN=NL
156 IF LK=0 THEN 214
157 PRINTLN,
158 PRINT#6,CHR$(1);CHR$(1);
159 PRINT#6,CHR$(LL);CHR$(LH);
160 X=1
161 GOSUB  231
162 V(X)=T
163 IF T>0 THEN X=X+1:GOTO 161
164 GOSUB  229
165 LK=T+T1
166 GOSUB 229
167 NL=T1+(256*T)
168 IF LK=0 THEN 171
169 LL=T1
170 LH=T
171 IF F THEN F=0:GOTO 155
172 REM *************************
173 REM *                       *
174 REM * RICONOSCIMENTO TOKENS *
175 REM *                       *
176 REM *************************
177 X=1
178 IF V(X)<>58 THEN 189
179 IF X=1 THEN 206
180 LN=LN+1
181 IF LN>=NL THEN 206
182 PRINT#6,CHR$(0);CHR$(1);CHR$(
    1);
183 H=INT(LN/256)
184 L=LN-(256*H)
185 PRINT#6,CHR$(L);CHR$(H);
186 X=X+1
187 IF V(X)=32 OR V(X)=58 THEN 18
    6
188 GOTO 178
189 IF V(X)<128 OR V(X)>155 THEN
    201
190 IF V(X)=128 OR V(X)>153 THEN
    198
191 IF V(X)<137 OR V(X)>144 THEN
    201
192 IF V(X)=140 OR V(X)=141 THEN
    201
193 REM *************************
194 REM *                       *
195 REM *    SCOMPATTAMENTO     *
198 PRINT#6,CHR$(V(X));
199 IF V(X)>0 THEN X=X+1:GOTO 198
200 GOTO 155
201 IF V(X)<>34 THEN 206
202 PRINT#6,CHR$(V(X));
203 X=X+1
204 IF V(X)=34 OR V(X)=0 THEN 206
205 GOTO 202
206 PRINT#6,CHR$(V(X));
207 IF V(X)>0 THEN X=X+1:GOTO 178
208 GOTO 155
209 REM *************************
210 REM *                       *
211 REM *  FINE SCOMPATTAMENTO  *
212 REM *                       *
213 REM *************************
214 PRINT#6,CHR$(0);CHR$(0)
215 CLOSE 5:CLOSE 6:CLOSE 15
216 PRINT"[CLEAR]":PRINT TAB(10):
    PRINT"[4 DOWN]*COMPATTAMENTO
     FINITO"
217 FOR K=1 TO 500:NEXTK
218 PRINT"[CLEAR]\L PROGRAMMA SCO
    MPATTATO E'STATO SALVATO"
219 PRINT"SUL DISCO NEL DRIVE PRE
    SCELTO."
220 INPUT "[5 DOWN]\UOI SCOMPATTA
    RE UN ALTRO PROGRAMMA([RVS]S/
    N[RVOFF])";SN$
221 PRINT"[CLEAR]"
222 IF SN$="S" THEN 132
223 SYS64738
224 REM *************************
225 REM *                       *
226 REM *      SUBROUTINES      *
227 REM *                       *
228 REM *************************
229 GOSUB 231
230 T1=T
231 GET #5,C$
232 GOSUB 236
233 IF C$="" THEN T=0:RETURN
234 T=ASC(C$)
235 RETURN
236 INPUT#15,EN,EM$,ET,ES
237 IF EN=0 THEN RETURN
238 PRINT"[CLEAR][RVS]¯RRORE NEL
    DISCO"
239 PRINT
240 PRINTEN;EM$;ET;ES
```

IL TRIANGOLO DI TARTAGLIA

di **Mauro Massetti**

Quando si affronta l'elevazione a potenza di un binomio del tipo (x + y) elevato a n;o (x-y) elevato a n, non si hanno problemi (almeno spero) sino ad esponenti minori o uguali a 4. Questo perchè nelle scuole, normalmente, questi sviluppi sono trattati solo sino a livelli inferiori, che, peraltro, sono i più usati.

Può però capitare di imbattersi in un elevamento a potenza di ordine superiore, ed ecco che... cominciano i dolori; questo accade soprattutto quando il binomio è del tipo (x-y) elevato a n. In questo caso, infatti, il segno dei vari coefficienti varia alternativamente e basta inver-

tire (malauguratamente) l'ordine dei fattori per ottenere risultati del tutto errati. A tal proposito ci viene incontro il triangolo dei fattori detto "triangolo di Tartaglia" dal matematico omonimo che ne scoprì le proprietà delle regole che lo governano.
Queste regole, invero, sono poche e non presentano difficoltà alcuna nel loro apprendimento, ma per essere spiegate con semplicità, necessitano di un esempio: consideriamo il binomio (x + y) elevato a n; per n = 0 si avrà come risultato per qualsiasi valore di x e y

1
per n = 1 1 x 1y
per n = 2 1x/2 2xy 1x/2
per n = 3 1x/3 3x/2y/1 3x1y/2 1y/3

ecc, per cui si nota subito la disposizione a triangolo dei fattori. Si possono notare due cose importanti che subito saltano all'occhio:

• se per ogni fattore si considera la coppia-prodotto delle variabili (xy), si nota che l'esponente decresce dal valore massimo allo zero per la prima e cresce da zero al valore massimo per la seconda;
• la somma degli esponenti delle variabili è costante ed è pari all'esponente del binomio;

Veniamo ora alla cosa più importante. Se pensiamo di riscrivere tutti e solo i coefficienti nella forma

0001000
00011000
000121000
0001331000
00014641000

possiamo notare che se si considerano tanti piccoli triangoli con i vertici del lato orrizzontale superiore in corrispondenza di due valori dei coefficienti e il vertice inferiore nella posizione occupata dal coefficiente da cercare, detto coefficiente ha valore pari alla somma degli altri due coefficienti appartenenti allo stesso triangolo.
A questo punto analizziamo più in dettaglio il programma per meglio comprendere la struttura. Dopo un rinvio al blocco della videata iniziale, di imputazione e di scelta, si accede al blocco principale del programma (riga 30). Come è facilmente rimarcabile dagli schemi sopra mostrati, in corrispondenza di esponenti pari ci sono un numero dispari di fattori, mentre per esponenti dispari il numero dei fattori è pari.
Si è operato quindi come segue: in caso di numero dispari dei fattori, il centrale viene considerato a parte e si pongono, invece, tutti gli altri (come un numero pari di fattori) su un vettore centrandone la posizione rispetto alla metà della lunghezza del vettore stesso. Questo perché per un qualsiasi vettore di lunghezza n è, nel nostro caso, x((n/2)-1) = x((n/2) + il) per qualsivoglia valore di I < n/2. Si possono ottenere ora i valori degli esponenti sommando fra loro quelli degli esponenti della riga superiore secondo le regole precedentemente esposte. La procedura sopra indicata non pone particolari problemi se il binomio considerato è del tipo (x + y) elevato a n. Se invece fosse del tipo (x-y) elevato a n, come già menzionato, il segno dei fattori cambia alternativamente (cominciando però sempre da positivo). In questo caso bisogna quindi prestare molta attenzione al coefficiente centrale (nel programma denominato con la variabile CC) in quanto risulta essere positivo per esponenti del binomio esattamente divisibili per 4 e negativo per esponenti divisibili solo per 2.
A causa dello spazio orizzontale limitato (40 colonne video) lo sviluppo a triangolo non viene visualizzato per esponenti > 9, mentre per quello in fattori non vi sono limitazioni (basta saper pazientemente attendere e cambiare eventualmente, i dimensionamenti dei vettori C⁻ e CR tenendo presente che deve essere DIM = (ES + 1).

```
100 REM ***********************
105 REM *                     *
110 REM *   IL TRIANGOLO DI   *
115 REM *  T A R T A G L I A  *
120 REM *         DI          *
125 REM *  MAURO MASSETTI     *
130 REM *                     *
135 REM ***********************
140 DIM CF(50),CR(50):GOSUB 575
145 GOTO 500:REM  *TESTA PROGRAMM
    A
150 REM ******************
155 REM *  CALCOLO DEI     *
160 REM *  COEFFICIENTI    *
165 REM * PER LO SVILUPPO *
170 REM ******************
175 FOR I=1 TO ES
180 IF ((I/2)-INT(I/2))<>0 THEN 2
    05
185 CC=CR(25)*2
190 FOR J=1 TO (ES/2)
195 CF(26-J)=CR(26-J)+CR(25-J):CF
    (25+J)=CF(26-J)
200 NEXTJ:GOTO 245
```

```
205 CF(25)=CC+CR(25):CF(26)=CF(25
    ):IF ES=1 THEN 245
210 FOR J=2 TO ((ES+1)/2)
215 CF(26-J)=CR(27-J)+CR(26-J):CF
    (25+J)=CF(26-J)
220 NEXTJ
225 REM ******************
230 REM *  COPIA MATRICE  *
235 REM *     CF SU CR     *
240 REM ******************
245 FOR J=1 TO 50:CR(J)=CF(J):NEX
    TJ:GOSUB 430
250 REM ***********************
255 REM * SVILUPPO IN FATTORI *
260 REM ***********************
265 NEXTI:KO=0:PRINT:PRINT
270 FOR I=1 TO 25
275 IF CF(I)=0 THEN 290
280 IF CF(I)=1 THEN PRINT"X↑";ES:
    KO=1:GOTO 290
285 GOSUB 380:GOSUB 405
290 NEXTI:IF SG$="-" AND ((ES/4)-
    INT(ES/4))<>0 THEN CC=-CC
295 IF ((ES/2)-INT(ES/2))<>0 THEN
     305
300 PRINTCC;"X↑";KO;"Y↑";KO:KO=KO
    +1:CF(25)=-CF(25)
305 FOR I=26 TO 50
310 IF CF(I)=0 THEN I=50:GOTO 330
315 GOSUB 380:IF CF(I)=1 THEN PRI
    NT"Y↑";ES:GOTO 330
320 IF CF(I)=-1 THEN PRINT"-Y↑";E
    S:GOTO 330
325 GOSUB 405
330 NEXTI
335 REM ****************
340 REM * CONTINUI S/N´*
345 REM ****************
350 INPUT "VUOI CONTINUARE(S/N)";
    S$
355 IF S$="S" THEN PRINT"[CLEAR]"
    ;"[HOME]":GOTO 145
360 END
365 REM ***********************
370 REM * SUBROU CAMBIO SEGNO *
375 REM ***********************
380 IF SG$="-" AND CF(I-1)>0 THEN
     CF(I)=-CF(I)
385 RETURN
390 REM *************************
395 REM * SUBROU STAMPA FATTORI *
400 REM *************************
405 PRINTCF(I);"X↑";ES-KO;"Y↑";KO
    :KO=KO+1
410 RETURN
415 REM *******************
420 REM * SUBROU TRIANGOLO *
425 REM *******************
430 IF I=1 THEN PRINT TAB(21);"1"
435 IF I>9 THEN 495
440 R$="":FOR J=21 TO 25
445 IF CF(J)=0 THEN 455
450 R$=R$+RIGHT$(("    "+STR$(CF(
    J))),4)
455 NEXTJ
460 IF ((I/2)-INT(I/2))<>0 THEN 4
    70
465 R$=R$+RIGHT$(("    "+STR$(CC)
    ),4)
470 FOR J=26 TO 30
475 IF CF(J)=0 THEN J=30:GOTO 485
480 R$=R$+RIGHT$(("    "+STR$(CF(
    J))),4)
485 NEXTJ
490 PRINT TAB(18-I*2);R$
495 RETURN
500 FOR I=1 TO 50:CF(I)=0:CR(I)=0
    :NEXTI:CC=1:SG$="+"
505 REM *******************
510 REM *    SVILUPPO DI    *
515 REM * BINOMIO ELEVATO *
520 REM *  A POTENZA E     *
525 REM *  TRIANGOLO DI    *
530 REM *    TARTAGLIA     *
535 REM *******************
540 PRINT"BINOMIO: A) (X+Y)↑N"
545 PRINT"         B) (X-Y)↑N"
550 PRINT:INPUT "FATE LA SCELTA(A
    /B)";S$
555 IF S$="B" THEN SG$="-"
560 INPUT "IMPOSTARE L´ESPONENTE
    N";ES
565 IF  ES=0 THEN PRINT"SVILUPPO
    = 1":GOTO 350
570 GOTO 175
575 FOR R=1 TO 10:PRINT:NEXTR
580 PRINT TAB(15);"TRIANGOLO":PRI
    NT TAB(19);"DI":PRINT TAB(15)
    ;"TARTAGLIA"
585 FOR R=14 TO 24:PRINT:NEXTR:PR
    INT TAB(10);"DI: MAURO MASSET
    TI"
590 FOR H=1 TO 5000:NEXTH:PRINT"[
    CLEAR]";"[HOME]":RETURN
```

PROGRAMMAZIONE STRUTTURATA

Programma SCONTO: (ingresso: tastiera; uscita: video) calcolo dello sconto da applicare ad un importo dato

di **Mariangela Guardione**

Questa serie di articoli, vuole trattare un argomento che riveste attualmente una grandissima importanza derivata da una diffusione sempre maggiore nella vita quotidiana degli home e personal computers.

Con il termine "programmazione" si intende una disciplina autonoma che tratta i metodi di formulazione e di costruzione degli algoritmi. Un algoritmo è una legge che governa un'intera classe di processi di elaborazione e controllo dei dati. Per questo motivo, deve essere creato a partire da unità logiche adeguate e sicure.

Perchè la programmazione abbia la struttura di un metodo rigoroso devono essere individuati i problemi e le tecniche che le sono proprie, e che devono essere quindi valide per ogni tipo di applicazione.

Infatti, fino al 1960, la programmazione veniva eseguita usando un metodo chiamato "codificazione" che consisteva in una traduzione minuziosa delle istruzioni in numeri binari, ottali o esadecimali che fornivano quindi il programma pronto per essere eseguito dal calcolatore.

In questo lavoro di codifica, il programmatore doveva adattare il programma alle caratteristiche del calcolatore utilizzato. Doveva quindi conoscere profondamente e dettagliatamente la macchina utilizzata ed era impossibilitato ad adattare i programmi a macchine diverse. Ogni centro di calcolo doveva mettere a punto i propri programmi e, nell'evenienza dell'acquisto di un nuovo calcolatore, doveva abbandonare tutti i programmi preparati precedentemente ricominciando quindi un nuovo lavoro di codificazione.

Inconvenienti

Tutto questo portava il programmato-

re a usare le conoscenze acquisite con l'esperienza per inventare "trucchi" sempre più sottili. Il tutto per creare dei programmi che però risultavano avere una struttura logica molto difficile, sia da verificare, che da utilizzare da parte di un altro programmatore. Il problema principale: questa logica non corrispondeva in alcun modo a quella umana. E' dall'insieme di tutti questi inconvenienti che sorse la necessità di strutturare la programmazione per costruire in modo sistematico i programmi e per renderli di facile e immediata comprensione. Nacquero quindi i linguaggi di programmazione ad alto livello, che rappresentano il formalismo con cui vengono espresse le istruzioni per una macchina ideale, costruita cioè a misura d'uomo.

Da ciò risulta evidente che un programma deve essere costituito da una sequenza di istruzioni che il calcolatore possa comprendere; e questo vuol dire che l'insieme delle istruzioni che formano il programma devono servire a specificare esattamente le azioni che devono essere eseguite. Questa esigenza ineliminabile di chiarezza e di esattezza costituisce la differenza principale fra la comunicazione con le macchine e quella tra gli uomini. Per poter lavorare con i calcolatori è necessario usare espressioni chiare e precise, in quanto l'ambiguità, in questo specifico campo, non è ammessa.

E' per questo motivo che gli "schemi di flusso" costituiscono un mezzo facilmente comprensibile e molto usato per rappresentare i programmi, in quanto forniscono, tramite una rappresentazione grafica e quindi immediata, un'immagine molto chiara di come una serie di azioni possano succedersi l'una dopo l'altra.

Infatti in questi diagrammi sono rappresentati due tipi di istruzioni: le assegnazioni (racchiuse nei rettangoli) e le decisioni (racchiuse nei rombi).

Ad una decisione possono seguire una o più istruzioni che indicano la possibilità di una scelta che può avvenire in un senso (+) o nell'altro (-), mentre una ripetizione viene indicata da un loop, come schematizzato in figura 1.

Un programma quindi fornisce le regole di comportamento che regolano un numero in genere imprecisato di processi. Questi singoli processi presentano lo stesso modello di comportamento, ma si differenziano per i valori assunti dalle diverse variabili in ogni istante. Da tutto ciò nasce la necessità di avere un metodo di prova analitico per la verifica dei programmi, che si astragga dalle proprietà dei singoli, per ottenere delle regole valide in generale.

Questo metodo è noto come "verifica dei programmi". In aggiunta a questo, è necessario che i vari programmi siano corredati da una buona documentazione che ne completi comprensione e utilizzo.

Storia della programmazione strutturata

La nascita, se così si può dire, della programmazione strutturata avvenne nel 1958 per opera di un gruppo internazionale di esperti che adottarono, per creare il primo di questi linguaggi (ALGOL), le idee originali di Rutishauser. Quest'ultimo espose nel 1952 con chiarezza l'idea di sviluppare un linguaggio che fornisse una rappresentazione semplice e naturale della programmazione e che parallelamente potesse essere usato direttamente per l'elaborazione.

Una programmazione così strutturata fu ispirata dal campo dell'analisi numerica, che divenne il punto di partenza per adottare la notazione formale della matematica. Un altro linguaggio introdotto negli anni '70 e che segnò la nascita ufficiale della programmazione strutturata fu il PASCAL. Bisogna però, a questo punto, tenere presente che per apprendere pienamente e fare propria la logica della programmazione strutturata è indispensabile essere consci dell'importanza di saper costruire strutturalmente bene i programmi, mentre risulta del tutto secondaria la loro codificazione.

Un caso: il Basic

L'utilizzo di questo linguaggio, nella sua forma primitiva, portava ad una certa insoddisfazione dal punto di vista dell'applicazione. In questa sede si vuole proporre un metodo per poter utilizzare in maniera adeguata questo linguaggio e per cercare, inoltre, di indirizzare gli utilizzatori del basic verso un'impiego della programmazione strutturata.

Si vuole altresì dimostrare che anche in questo linguaggio si può programmare in maniera da poter evidenziare l'aspetto metodologico che sta alla base della stesura dei programmi poichè questo, come abbiamo già più volte ripetuto, astrae, entro determinati termini, dal linguaggio di programmazione.

Il primo punto da affrontare, per una buona stesura del programma, consiste nell'analisi delle strutture fondamentali per il controllo del flusso logico.

Controllo del flusso

Per sviluppare questo concetto si suddivide il programma in una serie di fasi.

● **Definizione del problema.** In questa fase devono comparire gli enunciati chiari delle elaborazioni che si vogliono eseguire e dei dati che verranno utilizzati nella fase di calcolo. In pratica, tutto ciò rappresenta la descrizione in chiaro del problema che si vuole affrontare con l'ausilio dell'elaboratore. Esempio:

Dato l'ammontare di un prestito (importo) si vuole conoscere il valore dell'interesse sull'importo considerando che:

- se l'importo è > 10000000 di lire
- allora la percentuale di interesse è 10%
- altrimenti la percentuale di interesse è 8%

● **Analisi del problema.** Questa fase

presenta la descrizione dei dati in ingresso e in uscita e il modello di calcolo che viene utilizzato. Esempio:

- nome attribuito al programma: interesse
- dati in ingresso: importo
- dati in uscita: interesse su importo
- modello di calcolo:
  dati ausiliari: percint (percentuale di interesse)
  elaborazioni: se importo < 10000000 allora percint = 10
  altrimenti, percint = 8
  interesse = percint*importo

● **Descrizione dei dati dell'algoritmo.** Quest'ultima fase prevede la formulazione del programma in pseudocodifica, vale a dire che al posto delle istruzioni vere e proprie del linguaggio BASIC e dei dati effettivi si utilizzano delle espressioni del linguaggio comune per descrivere le operazioni da eseguire e i dati da memorizzare.

I diagrammi sintattici

Per poter meglio descrivere le fasi di un programma e le istruzioni del linguaggio BASIC si usa di norma un grafico molto rigoroso, ma di immediata comprensione: il "diagramma sintattico". Per meglio comprenderlo passiamo ad esaminare ad uno a uno i simboli che vi compaiono.

Simboli terminali: essi sono rappresentati graficamente nella forma:

oppure:

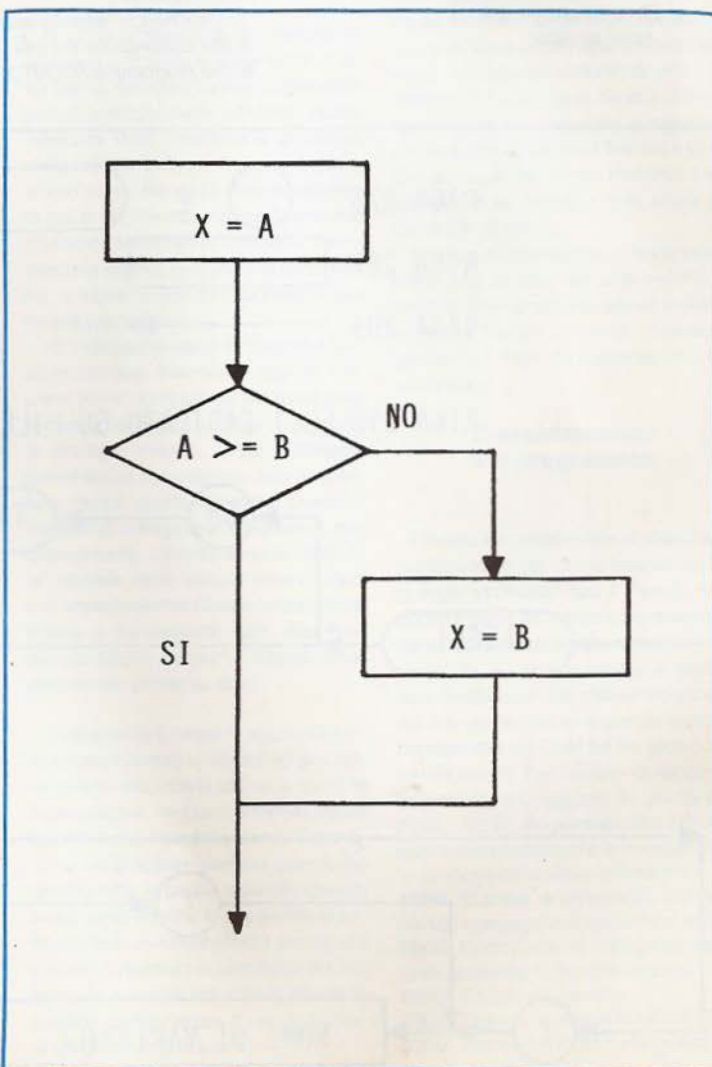
e vanno utilizzati così come appaiono non richiedendo alcuna ulteriore spiegazione.

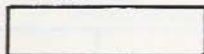
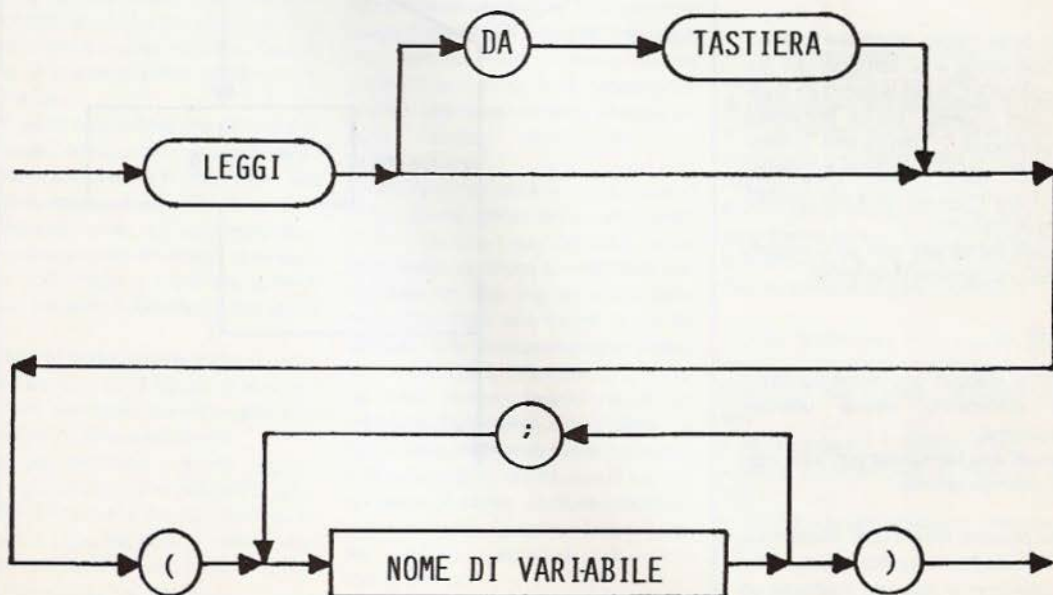
Simboli non terminali: essi sono rappresentati dal simbolo:

e non possono essere usati direttamente in quanto il loro significato deve essere ulteriormente spiegato mediante un altro diagramma o tramite una descrizione.

Linee di flusso: sono rappresentate da linee orientate che individuano l'ordine con cui vengono letti ed eseguiti i simboli nei digrammi sintattici. In un punto di diramazione l'utilizzatore può scegliere di seguire una qualunque via a meno che non esistano ulteriori limitazioni.

Impiegando questi diagrammi il lettore si renderà conto della loro potenza sia da un punto di vista didattico che come utile mezzo per rendere la programmazione più flessibile. A questo scopo si illustra, qui di seguito, un esempio che utilizza i diagrammi sintattici per descrivere una sequenza di operazioni.

Si carichi da tastiera una serie di variabili:

- Descrizione dei dati
  dati scambiati con l'esterno
  dati in ingresso
  IMPORTO
  % SCONTO
  dati in uscita
  IMPORTO SCONTATO
- *fine dati scambiati con l'esterno*
  dati ausiliari
  SCONTO SU IMPORTO
- *fine descrizione dei dati*

- Elaborazione dei dati
  scrivi su video
  <<RET> "calcola importo scontato" <RET>>
  scrivi su video
  ("ammontare e percentuale sconto" <RET> <RET>)
  scrivi su video
  ("fornire l'importo" <RET>)
  leggi da tastiera (IMPORTO)
  scrivi su video
  ("fornire la percentuale di sconto")
  leggi da tastiera (% SCONTO)
  SCONTO su IMPORTO=%SCONTO*IMPORTO/100
  IMPORTO SCONTATO=IMPORTO-SCONTO su IMPORTO
  scrivi su video
  <<RET> "importo scontato=", IMPORTO SCONTATO)
- *fine elaborazione dei dati*
- *fine programma SCONTO.*

COMPUTER MENACE

di **Marco De Rosa**

Menace. Ovvero: Computer Matchbox Educable Naughts and Crosses Engine.

La storia comincia nel 1961 ad Edimburgo, capitale della Scozia. In quegli anni un computer come il CBM 64 occupava pressappoco una stanza, pesava qualche tonnellata e non era certo alla portata delle tasche di un privato. Un certo Donald Michie, professore di biologia all'università, ebbe allora una idea eccezionale per costruire con poca spesa una macchina capace di apprendere delle semplici strategie. In un articolo in Penguin Science Survey del 1961, vol. 2, egli descrive come costruire una macchina che impara a giocare il Filetto (o Tris), usando circa trecento scatole di fiammiferi (!!), e una manciata di perline colorate.

Su ognuna delle scatole egli incollò delle possibili configurazioni raggiungibili in una partita di Tris, eliminando quelle banali, quelle irragiungibili, e le speculari, e contrassegnò ognuna delle mosse con un colore. Poi riempì ogni scatola con tante perline quante erano le mosse, ognuna del colore corrispondente. A questo punto cominciò a giocare con la macchina usando la seguente strategia:

- Si prende la scatolina corrispondente alla configurazione sulla scacchiera, si agita e si estrae casualmente una delle palline. Si esegue la mossa del colore della pallina estratta. Questa corrisponde alla mossa della macchina.
- L'errore umano risponde alla mossa, giocando nel modo migliore possibile.
- Si eseguono i passi 1 e 2 fino alla vittoria della macchina o dell'uomo.
- Si vince la macchina si rimettono le palline al loro posto e si comincia un'altra partita. Se vince l'uomo, si prende la pallina corrispondente all'ultima mossa effettuata dalla macchina e si elimina dalla scatola. Le altre vengono rimesse al loro posto. Nel caso abbastanza raro in cui si giunge ad una configurazione che corrisponde ad una scatola senza nessuna pallina, la macchina abbandona, e toglie quella corrispondente alla mossa precedente.

All'inizio ovviamente la macchina giocò in maniera totalmente stupida, ma, piano piano cominciò ad eliminare tutte le mosse perdenti e a conservare invece le strategie vincenti. Dopo diciassette partite aveva abbandonato tutte le aperture tranne quella d'angolo. Dopo la ventesima pareggiava abbastanza frequentemente, e Michie tentò di "distrarla" usando delle varianti fallaci. Dopo uno sbandamento iniziale la macchina imparò a fronteggiarle tutte. Alla fine, quando Michie decise di ritirarsi, essa vinceva otto partite su dieci.

Ovviamente il tempo di apprendimento è proporzionale all'abilità del giocatore umano, dato che le palline, e quindi le mosse cattive, vengono eliminate solo in caso in cui la macchina perde. Essa diventa un giocatore perfetto quando ha giocato tutte le partite possibili. Questo ovviamente elimina la possibilità di costruzione di un Robot siffatto, per i giochi che siano appena più complessi del Tris (pensate a quante potrebbero essere le possibili configurazioni di una scacchiera nel gioco degli scacchi!).

Nel 1984 è più comodo implementare una versione di Menace (macchina di scatolette di cerini educabile al gioco del Tris), su un personal computer che abbia un minimo di RAM per poter tenere un "vocabolario" di configurazioni.

Come si usa Menace

Dopo aver caricato il programma e aver dato il RUN, vi viene chiesto se volete caricare un vocabolario già esistente sul disco. Dopo aver risposto con S o N, vi viene presentata la scacchiera vuota. Voi giocate sempre per primi, indicando il numero della casella come in figura 2. Menace risponde automaticamente e si accorge della fine della partita in qualsiasi occasione. Può anche abbandonare se ritiene di non avere più possibilità di vittoria.

Il tempo di risposta con un vocabolario pieno può arrivare ad una trentina di secondi. Premendo in qualsiasi momento il tasto "freccia a sinistra", potete registrare sul disco gli aggiornamenti del vocabolario.

Considerazioni sul programma

Questo è il classico tipo di idea che si realizzerebbe benissimo usando un linguaggio strutturato tipo il Pascal. Purtroppo il Basic ha il difetto di essere residente sulla macchina al momento dell'acquisto, e i programmatori in genere sono troppo pigri per infilare il dischetto del Pascal nel Driver e per caricarlo (e pensate che sul CBM 64 ne girano ben tre versioni!!). Per cercare di rendere il programma più leggibile in un mare di FOR... NEXT, ho pensato allora di creare otto subroutines che si occupano della gestione delle varie operazioni:

1000 Stampa la schermata comprendente i messaggi e la scacchiera attuale.
2000 Controlla se la configurazione attuale presenta delle righe vincenti.
3000 Decide chi ha vinto.
4000 Esegue la mossa di MENACE.
5000 Permette l'input della mossa del-

lo sfidante umano.

6000 Gestisce la fine della partita.

7000 Salva sul disco il vocabolario delle configurazioni aggiornato all'ultima mossa.

8000 Carica dal disco il vocabolario.

Queste vengono poi gestite da un programma principale lungo una ventina di righe, estremamente semplice e descritto da parecchie REM. L'unica subroutine un po' complessa è la 4000, di cui potete vedere il diagramma di flusso in figura 1.

E' possibile compilare il programma per aumentare la velocità di risposta. I risultati migliori li ho ottenuti usando l'Austro Compiler piuttosto che il Pet Speed 64.

Descrizione delle Variabili

M(9,500) Contiene il vocabolario di tutte le configurazioni trovate fino ad ora dal programma. Il primo indice è relativo alla casella (dall'1 a 9), il secondo alla configurazione.

P(9) Contiene la configurazione attuale della scacchiera.

PP(9,9) Contiene tutte le configurazioni trovate nella partita corrente. Il primo indice è relativo alla casella (dall'1 a 9), il secondo alla configurazione.

PU(9) Contiene i puntatori, per ogni mossa, alla relativa configurazione nel vocabolario. Zero indica la creazione di una nuova matrice nel vocabolario.

PT(9) Contiene tutte le mosse della partita.

P$(9) Contiene la configurazione attuale della scacchiera, in versione stampabile sullo schermo.

In tutti i vettori di configurazione, il numero uno indica le mosse umane, il due le mosse del computer, il tre le mosse non eseguibili da Menace, lo zero quelle possibili per entrambi.

P1 Numero di configurazioni presenti nel vocabolario.

MO Mossa attuale.

K Contatore del numero delle mosse.

FL Flag di vittoria. Il numero uno indica la vittoria umana, il due quella di Menace, il tre l'abbandono dell'elaboratore, lo zero la patta.

Tutte le altre sono variabili d'appoggio o di ciclo.

Descrizione del programma

15 Mette il colore di schermo e di sfondo sul verde.

17 Inizializza la funzione Random.

20 Pulisce lo schermo e mette il colore dei caratteri sul bianco.

30 Dimensiona i vettori.

35-37 Chiede se si vuole caricare un vocabolario già esistente sul disco. Se la risposta è sì, va alla **subroutine 8000,** altrimenti continua.

38 Pulisce lo schermo.

40 Stampa su schermo scacchiera e messaggi usando la **subroutine 1000.**

50 Accetta la mossa dell'essere umano, usando la **subroutine 5000.**

60 Come la **40.**

65 Controlla se uno dei due contendenti ha vinto, usando le **subroutine 2000 e 3000.**

70 Controlla se sono state eseguite nove mosse e se il flag di vittoria è nullo. In caso positivo considera conclusa la partita con risultato di parità, chiama la **subroutine 6000** (fine partita), e torna alla **riga 40.** Altrimenti continua.

80 Controlla se il flag di vittoria è 1 o 2. In caso positivo considera conclusa la partita, chiama la **subroutine 6000** (fine partita) e torna alla **riga 40.** Altrimenti continua.

90 Esegue la mossa di Menace usando la **subroutine 4000.**

95 Controlla se il flag di vittoria è uguale a tre (abbandono). In caso positivo considera conclusa la partita, chiama la **subroutine 6000** (fine partita) e torna alla **riga 40.** Altrimenti continua.

100 Come la **riga 40.**

110 Come la **riga 65.**

120 Come la **riga 80.**

130 Torna alla **riga 40.** Con questa riga si conclude il corpo principale del programma.

1000-1002 Stampa messaggi sullo schermo.

1005-1007 Prende la configurazione attuale contenuta in P(I) e la converte in vettore stringa colorando in blu e rosso i numeri 1 e 2.

1010-1060 Stampa sullo schermo la scacchiera con alcuni messaggi.

2000-2070 Controlla se esiste una riga verticale vincente.

2075-2140 Controlla se esiste una riga orizzontale vincente.

2150-2170 Controlla se esiste una riga diagonale vincente.

3000-3020 Questa subroutine viene chiamata tre volte dalla **2000.** In caso di vittoria di uno dei giocatori stampa il messaggio corrispondente e setta il flag di vittoria in modo appropriato.

4000-4516 Vedi il diagramma di flusso in figura 1.

5000-5021 Chiede la mossa umana.

5022 Controlla se è stato premuto il tasto "freccia a sinistra". In caso positivo salva sul disco il vocabolario attuale usando la **subroutine 7000** e torna alla **5020.** In caso contrario continua.

5025 Controlla che la mossa sia lecita. Se non lo è torna alla **5020.** Se lo è continua.

5027 Aggiorna il contatore delle mosse e la matrice di configurazione attuale.

5030-5031 Archivia la configurazione attuale nel vettore PP(9,9).

6000-6010 Controlla il flag di vittoria. Se ha vinto Menace o c'è una patta salta alla **6050** senza aggiornare il vocabolario. Infatti Menace impara solo in caso di sconfitta.

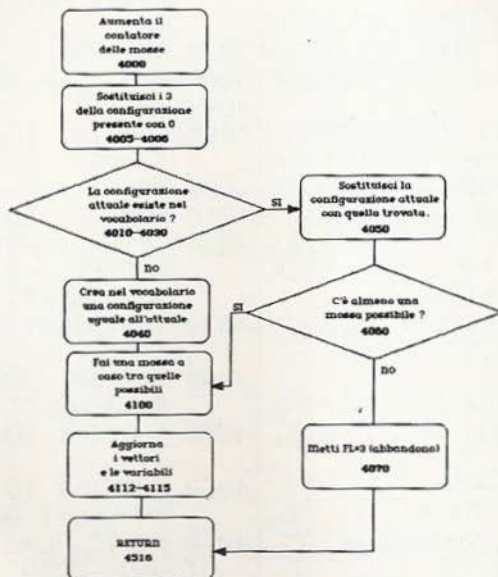
6015 Diminuisce il contatore delle mosse per tornare all'ultima mossa eseguita da Menace.

6016 Nel caso in cui ci sia un abbandono o una patta, diminuisce il contatore delle mosse ulteriormente in una unità.

6030 Mette un 3 nella configurazione corrispondente all'ultima mossa di Menace, aggiornando il vocabolario.

6050-6080 Pulisce le variabili per la nuova partita.

7000-7080 Crea sul disco un file sequenziale di nome DATA, in cui salva il vocabolario, preceduto dal numero di configurazioni presenti nello stesso.

8000-8080 Carica dal disco il file sequenziale DATA.

```
10 REM **************************
   ********
11 REM  COMPUTER MENACE BY MARCO
   DE ROSA
12 REM **************************
   ********
15 POKE 53280,5:POKE 53281,5
17 M=RND(0)
20 PRINT"[CLEAR][BIANCO]"
30 DIM M(9,500),P(9),PP(9,9),PU(
   9),PT(9),P$(9)
35 PRINT"DATI DA DISCO ?"
36 GET A$:IF A$="" THEN 36
37 IF A$="S" THEN GOSUB 8000
38 PRINT"[CLEAR]"
40 GOSUB 1000:REM  DISEGNA SCACC
   HIERA
50 GOSUB 5000:REM  MOSSA UOMO
60 GOSUB 1000:REM  DISEGNA SCACC
   HIERA
65 GOSUB 2000:REM  CONTROLLO VIT
   TORIA
70 IF K=9 AND FL=0 THEN PRINT"[D
   OWN]PATTA":GOSUB 6000:GOTO 40
   :REM  CONTROLLO PATTA
```

```
  80 IF FL=1 OR FL=2 THEN GOSUB 60
     00:GOTO 40:REM  FINE PARTITA
  90 GOSUB 4000:REM  MOSSA COMPUTE
     R
  95 IF FL=3 THEN GOSUB 6000:GOTO
     40:REM  FINE PARTITA PER ABBA
     NDONO
 100 GOSUB 1000:REM  DISEGNA SCACC
     HIERA
 110 GOSUB 2000:REM  CONTROLLO VIT
     TORIA
 120 IF FL=1 OR FL=2 THEN GOSUB 60
     00:GOTO 40:REM  FINE PARTITA
 130 GOTO 40:REM  FINE CORPO PROGR
     AMMA
1000 REM  SUB DISEGNA SCACCHIERA
1002 PRINT"[HOME][RIGHT][BLEU][2 R
     IGHT]MARCO DE ROSA - COMPUTER
      MENACE    [2 DOWN][BIANCO]"
1005 FOR I=1 TO 9:P$(I)=STR$(P(I))
     +" ":IF P(I)=1 THEN P$(I)="[R
     OSSO]"+P$(I)+"[BIANCO]"
1006 IF P(I)=2 THEN P$(I)="[BLEU]"
     +P$(I)+"[BIANCO]"
1007 NEXTI
1010 PRINTP$(1);"|";P$(2);"|";P$(3
     );"        CI SONO ";P1;" MATRI
     CI"
1020 PRINT"---+---+---"
1030 PRINT P$(4);"|";P$(5);"|";P$(
     6);"        NEL VOCABOLARIO"
1040 PRINT"---+---+---"
1050 PRINT P$(7);"|";P$(8);"|";P$(
     9);"        [ROSSO]1=UMANO[BIAN
     CO]  [BLEU]2=COMPUTER[BIANCO]
     "
1060 RETURN
2000 REM  SUB CONTROLLO VITTORIA
2010 FOR  I=1  TO  7  STEP  3
2015 S$=""
2020 FOR  J=I  TO  I+2
2030 S$=S$+STR$(P(J))
2040 NEXT J
2050 GOSUB  3000
2070 NEXT I
2075 FOR  I=1  TO  3
2080 S$=""
2090 FOR  J=I  TO  I+6  STEP  3
2100 S$=S$+STR$(P(J))
2110 NEXT J
2130 GOSUB  3000
2140 NEXT I
2150 S$=STR$(P(1))+STR$(P(5))+STR$
     (P(9)):GOSUB  3000
2160 S$=STR$(P(3))+STR$(P(5))+STR$
     (P(7)):GOSUB .3000
2170 RETURN
3000 REM  SUB SUB CONTROLLO
3005 IF  S$=" 2 2 2"  THEN  PRINT:
     PRINT"HO VINTO IO":FL=2:RETUR
     N
3010 IF  S$=" 1 1 1"  THEN  PRINT:
     PRINT"HAI VINTO TU":FL=1:RETU
     RN
3020 RETURN
4000 REM  SUB MOSSA COMPUTER
4003 K=K+1
4005 FOR I=1 TO 9:IF P(I)=3 THEN P
     (I)=0
4006 NEXTI
4010 FOR J=1 TO P1:FOR I=1 TO 9
4020 IF (P(I)=M(I,J) OR (M(I,J)=3
     AND P(I)=0)) THEN NEXTI:KK=J:
     J=P1:NEXTJ:GOTO 4050
4030 I=9:NEXTI:NEXTJ:REM  NON TROV
     ATA
4040 P1=P1+1:FOR I=1 TO 9:M(I,P1)=
     P(I):NEXTI:GOTO 4100
4050 FOR I=1 TO 9:P(I)=M(I,KK):NEX
     TI
4060 FOR I=1 TO 9:IF P(I)=0 THEN I
     =9:NEXTI:GOTO 4100
4070 NEXTI:FL=3:PRINT"[DOWN]ABBAND
     ONO":GOTO 4516
4100 REM  COMMON AREA
4110 MO=INT(RND(1)*9)+1:IF P(MO)<>
     0 THEN 4110
4112 P(MO)=2:PU(K)=MO:PT(K)=KK:KK=
     0
4115 FOR I=1 TO 9:PP(I,K)=P(I):NEX
     TI
4516 RETURN
5000 REM  SUB MOSSA UOMO
5010 PRINT"[5 DOWN]NUMERO ?";
5020 GET A$:IF A$="" THEN 5020
5021 MO=VAL(A$)
5022 IF A$="←" THEN GOSUB 7000:GOT
     O 5020
5025 IF MO<1 OR MO>9 OR P(MO)=1 OR
      P(MO)=2 THEN 5020
5027 P(MO)=1:K=K+1
5030 FOR I=1 TO 9:PP(I,K)=P(I):NEX
     TI
5031 RETURN
```

```
6000 REM  SUB FINE PARTITA
6010 IF FL=2 OR FL=0 THEN 6050
6015 K=K-1
6016 IF FL=3 THEN FL=0:K=K-1
6030 M(PU(K),PT(K))=3
6050 FOR I=1 TO 9:P(I)=0:PU(I)=0:P
     T(I)=0:NEXTI
6060 FOR I=1 TO 9:FOR J=1 TO 9:PP(
     I,J)=0:NEXTJ,I
6070 K=0:FL=0
6075 PRINT"[CLEAR]"
6080 RETURN
7000 REM  SUB SALVA FILE SU DISCO
7010 OPEN 2,8,2,"@0:DATA,S,W"
7020 PRINT#2,P1
7030 FOR J=1 TO P1
7040 FOR I=1 TO 9
7050 PRINT#2,M(I,J)
7060 NEXTI,J
7070 CLOSE 2
7080 RETURN
8000 REM  SUB LEGGI FILE DA DISCO
8010 OPEN 2,8,2,"0:DATA,S,R"
8020 INPUT#2,P1
8030 FOR J=1 TO P1
8040 FOR I=1 TO 9
8050 INPUT#2,M(I,J)
8060 NEXTI,J
8070 CLOSE 2
8080 RETURN
```

```
-------*-------------------------------- - - -
VAR.   ! LINEA DEL PROGRAMMA
-------*-------------------------------- - - -
       !
A$     !    36    37  5020  5021  5022
FL     !    70    80    95   120  3005  3010
       !  4070  6010  6016  6070
I      !  1005  1006  1007  2010  2020  2070
       !  2075  2090  2140  4005  4006  4010
       !  4020  4030  4040  4050  4060  4070
       !  4115  5030  6050  6060  7040  7050
       !  7060  8040  8050  8060
J      !  2020  2030  2040  2090  2100  2110
       !  4010  4020  4030  6060  7030  7050
       !  7060  8030  8050  8060
K      !    70  4003  4112  4115  5027  5030
       !  6015  6016  6030  6070
KK     !  4020  4050  4112
M      !    17
M(     !    30  4020  4040  4050  6030  7050
       !  8050
MO     !  4110  4112  5021  5025  5027
P$(    !    30  1005  1006  1010  1030  1050
P(     !    30  1005  1006  2030  2100  2150
       !  2160  4005  4020  4040  4050  4060
       !  4110  4112  4115  5025  5027  5030
       !  6050
P1     !  1010  4010  4020  4040  7020  7030
       !  8020  8030
PP(    !    30  4115  5030  6060
PT(    !    30  4112  6030  6050
PU(    !    30  4112  6030  6050
S$     !  2015  2030  2080  2100  2150  2160
       !  3005  3010
```

SPEEDLOAD / SAVE

di **Ernesto Sidoti**

Hai mai salvato blocchi di memoria? Magari di diversi Kbyte? Se è così questo programma può interessarti.

Spesso accade di dover archiviare su periferica valori che risiedono da una locazione ad un'altra, e magari il banco di memoria da copiare è di diverse migliaia di locazioni. Il metodo più intuitivo è senz'altro quello di avviare un ciclo che legga i valori nelle locazioni interessate e li scriva per mezzo delle istruzioni PRINT≠ e GET≠ su disco o su nastro. Un simile procedimento è semplice e veloce da realizzare, ma è molto lento nell'esecuzione che a volte può richiedere addirittura decine di minuti se le archiviazioni o le letture sono molte. Per abbreviare i tempi di attesa, puoi utilizzare delle routine già presenti sul tuo COMMODORE. Queste routine fanno parte del kernal.

Il programma proposto questo mese sfrutta questa tecnica e utilizza la SETLFS, la SETNAM, la SAVE e la LOAD. Questo programma ti permette di salvare su disco o su cassetta qualsiasi banco di memoria e viceversa di caricare su qualsiasi banco di memoria i valori archiviati su memoria di massa.

Descrizionc del programma

Linea 1100. Salto alla routine che realizza la maschera di apertura programma.

Linea 1170-1180. Calcolo del high-byte e del low-byte dell'indirizzo di par-

Linea 1380. Metto nella locazione 251 e 252 il low-byte e l'high-byte dell'indirizzo di partenza. Queste due locazioni, insieme alla 253 e alla 254, costituiscono un'area di 4 byte liberi per programmi realizzati dall'utente.

Linea 1390. Metto nell'accomulatore (POKE 780) il valore 251 che rappresenta l'indirizzo dove sono allocati i byte che indicano la locazione di partenza del banco da caricare.
Metto ancora nel registro x (POKE 781) il valore del low-byte e nel registro y (POKE 782) il valore dello high-byte dell'indirizzo dell'ultimo byte da tenza del banco su cui operare.

Linea 1190. Se si è scelto DISK, allora il programma continua alla **linea 1910.**

Linea 1200. Se si è scelto LOAD, il programma continua alla **linea 1530.**

Linee 1270-1280. Input dell'ultima locazione da salvare incrementato di 1 per salvare anche 1 byte successivo a quello dato.

Linee 1350-1390. Calcolo del high-byte e del low-byte dell'indirizzo di fine del banco da copiare.

Linea 1970. Salta alla **linea 1690** e apre il canale del tape.

```
C R O S S   R E F E R E N C E

PROGRAMMA : SPEEDLOAD/SAVE

-------*------------------------- - - -
VAR.   ! LINEA DEL PROGRAMMA
-------*------------------------- - - -
       !
BL     !  1360  1390  2010  2200
C$     !  1200  1910  2130  2140  2880  2890
       !  2900  2910  3000  3010  3020
EB     !  1350  1360  1390  2000  2010  2200
EM$    !  2090  2110  2310  2320
EN     !  2090  2100  2110  2310  2320
ES     !  2090  2110  2310  2320
ET     !  2090  2110  2310  2320
GY     !  2590  2610  2630  2650  2760  2780
       !  2800  2820
HH     !  1170  1180  1380  1540  2190  2330
J      !  2440  2450  2470  2500  2510  2520
       !  2540  2550  2560
K$     !  1190  2710  2720  2730  2740
L      !  1690  1700  1720  1810
LI     !  1180  1380  1540  2190  2330
NM$    !  1690  1730  2080  2150  2300  2920
S      !  1710  1730
ST     !  1600
T      !  2960
Y      !  2950
Z      !  1170  1180  1270  1280  1350  1360
       !  1720  1730  1980  1990  2000  2010
       !  2930
```

caricare.

Linea 1450. In questa linea richiamo la routine per il SAVE del kernal. Questa, prima di andare in esecuzione, leggerà i valori nei registri appena menzionati e si comporterà di conseguenza.

Linee 1530-1620. Legge un blocco dal tape.

Linea 1530. Apre il canale del registratore.

Linea 1540. Metto nella locazione 780 il valore che specifica il LOAD cioè lo 0 (1 se verify), nel registro x e nel registro y, il low-byte e l'high-byte dell'indirizzo dal quale comincerà il computer a caricare i valori provenienti dalla memoria di massa.

Linea 1590. Mando in esecuzione la routine del kernal per il load.

Linea 1600. Realizzo un test sulla variabile ST. Se questa variabile assume valore 48, ossia hanno valore 1 i bit 4 e 5, il file non è stato letto correttamente e il programma finisce.

Linee 1690-1880. Con questa routine apro il canale del tape in modo simile all'istruzione OPEN 1,1,0,NM$.

Linea 1690. Calcolo la lunghezza del nome del file.

Linea 1700. Metto il valore calcolato nella locazione 183 (in questa locazione va sempre messo il numero di lettere che compongono il nome del file con il quale si sta operando).

Linea 1710. Calcolo dell'indirizzo di fine array.

Linee 1720-1740. Metto il nome del file nelle locazioni appena dopo gli array.

Linea 1750. Apro il file logico mettendo nella locazione 780 il numero del file, nella 781 il numero della device e nella 782 l'indirizzo secondario.

Linea 1800. Chiamo la routine SETLFS del kernal.

Linea 1810. Metto nella locazione 780 il valore della lunghezza del nome del file e nella 781 e nella 782 l'indirizzo del puntatore fine array e mando in esecuzione la routine SETMAN del kernal.

Linea 1870. Avvio il registratore portando a valore 1 il settimo byte della locazione 157 (POKE 157, 128).

Linea 1980. Input dell'indirizzo di fine del banco di memoria da salvare.

Linee 2000-2010. Calcolo dell'high-byte e del low-byte.

Linea 2030. Controllo se esistono errori. Se così non è, salto alla **linea 2350.**

Linea 2110. Se l'errore è di tipo 63, allora sul disco esiste un file con lo stesso nome e quindi se si vuole scrivere un altro file su disco con il medesimo nome bisogna cancellare quello già esistente.

Linea 2150. Cancello il vecchio file.

Linee 2190-2210. Salvo il file su disco usando il medesimo procedimento descritto per il tape.

Linee 2290-2370. Leggo dal floppy.

Linee 2440-3040. Realizzo la maschera con le opzioni offerte dal programma.

Il programma descritto con qualche piccola modifica di ordine grafico può senza problemi essere usato sul VIC 20. Buon lavoro!.

ROUTINE	INDIRIZZO
LOAD	65493
SAVE	65496
SETNAM	65469
SETLFS	65466

Figura 1

```
1000 REM  **********************
1010 REM  *                    *
1020 REM  *  SPEEDSAVE/LOAD    *
1030 REM  *                    *
1040 REM  *        DI          *
1050 REM  *                    *
1060 REM  *  ERNESTO SIDOTI    *
1070 REM  *                    *
1080 REM  **E*****************S**
1090 :
1100 GOTO 2440
1110 REM  ----------------------
1120 REM  -  CALCOLO HIGH BYTE  -
1130 REM  -     E  LOW BYTE     -
1140 REM  -   DELL' INDIRIZZO   -
1150 REM  -     DI PARTENZA     -
1160 REM  ----------------------
1170 HH=INT(Z/256)
1180 LI=Z-HH*256
1190 IF K$="D" THEN 1910
1200 IF C$="L" THEN 1530
1210 :
1220 REM  **********************
1230 REM  *                    *
1240 REM  *  SALVA SUL TAPE    *
1250 REM  *                    *
1260 REM  **********************
1270 INPUT "[BLEU][DOWN]END ADDRES
     S    [BIANCO]";Z
1280 Z=Z+1
1290 REM  ----------------------
1300 REM  -  CALCOLO HIGH BYTE  -
1310 REM  -     E  LOW BYTE     -
1320 REM  -   DELL' INDIRIZZO   -
1330 REM  -       DI  FINE      -
1340 REM  ----------------------
1350 EB=INT(Z/256)
1360 BL=Z-EB*256
1370 GOSUB 1690
1380 POKE 251,LI:POKE 252,HH
1390 POKE 780,251:POKE 781,BL:POKE
```

```
    782,EB
1400 PRINT
1410 REM  ----------------------
1420 REM  - ROUTINE  DEL KERNAL -
1430 REM  -    PER  IL  SAVE    -
1440 REM  ----------------------
1450 SYS65496
1460 GOTO 1610
1470 :
1480 REM  **********************
1490 REM  *                    *
1500 REM  *   LEGGE DAL TAPE   *
1510 REM  *                    *
1520 REM  **********************
1530 GOSUB  1690
1540 POKE 780,0:POKE 781,LI:POKE 7
     82,HH
1550 REM  ----------------------
1560 REM  - ROUTINE  DEL KERNAL -
1570 REM  -    PER  IL  LOAD    -
1580 REM  ----------------------
1590 SYS65493
1600 IF (ST  AND  48) THEN PRINT"[
     DOWN] LOAD":PRINT"ERROR"
1610 CLOSE 1
1620 END
1630 :
1640 REM  **********************
1650 REM  *                    *
1660 REM  * APRE IL CANALE TAPE *
1670 REM  *                    *
1680 REM  **********************
1690 L=LEN(NM$)
1700 POKE 183,L
1710 S=256*PEEK(50)+PEEK(49)
1720 FOR Z=1  TO  L
1730 POKE S+Z-1,ASC(MID$(NM$,Z,1))
1740 NEXT
1750 POKE 780,1:POKE 781,1:POKE 78
     2,0
1760 REM  ----------------------
1770 REM  - ROUTINE  DEL KERNAL -
1780 REM  -        SETLFS       -
1790 REM  ----------------------
1800 SYS65466
1810 POKE 780,L:POKE 781,PEEK(49):
     POKE 782,PEEK(50)
1820 REM  ----------------------
1830 REM  - ROUTINE  DEL KERNAL -
1840 REM  -        SETNAM       -
1850 REM  ----------------------
1860 SYS65469
1870 POKE 157,128
1880 RETURN
1910 IF C$="L" THEN 2290
1920 :
1930 REM  **********************
1940 REM  *                    *
1950 REM  *   SALVA SUL DISCO  *
1960 REM  *                    *
1970 REM  **********************
1980 INPUT "[DOWN][BLEU]END ADDRES
     S      [BIANCO]";Z
1990 Z=Z+1
2000 EB=INT(Z/256)
2010 BL=Z-EB*256
2020 PRINT"[UP][BLEU]"
2030 OPEN 15,8,15,"I0"
2040 REM  ----------------------
2050 REM  - APERTURA DI UN FILE -
2060 REM  -  PER  LA SCRITTURA  -
2070 REM  ----------------------
2080 OPEN 3,8,1,"0:"+NM$+",P,W"
2090 INPUT#15,EN,EM$,ET,ES
2100 IF EN=0 THEN 2190
2110 IF EN<>63 THEN PRINT"[ROSSO][
     LEFT][DOWN]"EN;EM$;ET;ES:GOTO
      2350
2120 PRINT"[DOWN]FILE EXISTS. [BLE
     U]REPLACE (Y/N) ? [BIANCO]"
2130 GET C$:IF C$="" THEN 2130
2140 IF C$<>"Y" THEN 2350
2150 PRINT#15,"S0:"+NM$+",P,W"
2160 CLOSE 15
2170 CLOSE 3
2180 GOTO 2030
2190 POKE 157,128:POKE 251,LI:POKE
      252,HH
2200 POKE 780,251:POKE 781,BL:POKE
      782,EB
2210 SYS65496
2220 GOTO 2350
2230 :
2240 REM  **********************
2250 REM  *                    *
2260 REM  *   LEGGE DA DISCO   *
2270 REM  *                    *
2280 REM  **********************
2290 OPEN 15,8,15,"I0"
2300 OPEN 3,8,0,"0:"+NM$+",P,R"
2310 INPUT#15,EN,EM$,ET,ES
2320 IF  EN THEN PRINT"[ROSSO][DOW
     N]"EN,EM$;ET;ES:GOTO 2350
2330 POKE 157,128:POKE 185,0:POKE
```

```
     760,0:POKE 781,LI:POKE 782,HH
2340 SYS65493
2350 CLOSE 3
2360 CLOSE 15
2370 GOTO 2950
2380 :
2390 REM  **********************
2400 REM  *                    *
2410 REM  *       MENU'        *
2420 REM  *                    *
2430 REM  **********************
2440 J=5:PRINT"[CLEAR][UP]":POKE 5
     3280,12:POKE 53281,12
2450 PRINT TAB(J)"[BLEU]||||[GIALL
     O] SPEED-SAVE/LOAD (C) [BLEU]
     ||||
2460 PRINT
2470 PRINT TAB(J)"[GIALLO]       DI
     ERNESTO SIDOTI
2480 PRINT
2490 PRINT
2500 PRINT TAB(J)"[BLEU]     ┌─────
     ┐      ┌──────┐
2510 PRINT TAB(J)"     | DISK |
     | TAPE |
2520 PRINT TAB(J)"     └──────┘
     └──────┘
2530 PRINT
2540 PRINT TAB(J)"     ┌──────┐
     ┌──────┐
2550 PRINT TAB(J)"     | SAVE |
     | LOAD |
2560 PRINT TAB(J)"     └──────┘
     └──────┘
2570 :
2580 PRINT"[BIANCO][HOME][7 DOWN][
     10 RIGHT][RVS]DISK
2590 FOR GY=1 TO 90:NEXT
2600 PRINT"[HOME][7 DOWN][10 RIGHT
     ][RVOFF]DISK
2610 FOR GY=1 TO 70:NEXT
2620 PRINT"[HOME]":PRINT TAB(25)"[
     6 DOWN][RVS]TAPE
2630 FOR GY=1 TO 90:NEXT
2640 PRINT"[HOME]":PRINT TAB(25)"[
     6 DOWN][RVOFF]TAPE
2650 FOR GY=1 TO 70:NEXT
2660 REM  **********************
2670 REM  *                    *
2680 REM  * SCELTA DISCO O TAPE *
2690 REM  *                    *
2700 REM  **********************
2710 GET K$:IF K$="" THEN 2580
2720 IF K$<>"T" AND K$<>"D" THEN 2
     580
2730 IF K$="T" THEN PRINT TAB(25)"
     [HOME][7 DOWN][10 RIGHT][BLEU
     ]DISK"
2740 IF K$="D" THEN PRINT TAB(25)"
     [HOME][7 DOWN][25 RIGHT][BLEU
     ]TAPE"
2750 PRINT"[BIANCO][HOME][11 DOWN]
     [12 RIGHT][RVS]SAVE
2760 FOR GY=1 TO 90:NEXT
2770 PRINT"[HOME][11 DOWN][12 RIGH
     T][RVOFF]SAVE
2780 FOR GY=1 TO 90:NEXT
2790 PRINT"[BIANCO][HOME][11 DOWN]
     [23 RIGHT][RVS]LOAD
2800 FOR GY=1 TO 90:NEXT
2810 PRINT"[HOME][11 DOWN][23 RIGH
     T][RVOFF]LOAD

2820 FOR GY=1 TO 90:NEXT
2830 REM  **********************
2840 REM  *                    *
2850 REM  * SCELTA LOAD O SAVE *
2860 REM  *                    *
2870 REM  **********************
2880 GET C$:IF C$="" THEN 2750
2890 IF C$<>"S" AND C$<>"L" THEN 2
     750
2900 IF C$="L" THEN PRINT"[BLEU][U
     P][12 RIGHT]SAVE"
2910 IF C$="S" THEN PRINT"[BLEU][U
     P][23 RIGHT]LOAD"
2920 INPUT "[3 DOWN]FILENAME
       [BIANCO]";NM$
2930 INPUT "[DOWN][BLEU]STARTING A
     DDRESS [BIANCO]";Z
2940 GOTO 1170
2950 FOR Y=1 TO 2000:NEXT:PRINT"[H
     OME][14 DOWN]"
2960 FOR T=0 TO 8
2970 PRINT"
                      "
2980 NEXT
2990 PRINT"[HOME][BLEU][19 DOWN][1
     3 RIGHT]I BEGIN (Y/N) ?"
3000 GET C$:IF C$="" THEN 3000
3010 IF C$="Y" THEN 1100
3020 IF C$="N" THEN 3040
3030 GOTO 3000
3040 END
```

AUTORUN: COME COSTRUIRSELO

di **Giancarlo de Cobelli**

Per esaudire tutte le richieste che sono state rivolte alla nostra redazione per quanto riguarda la protezione dei programmi abbiamo deciso di pubblicare questo articolo che tratta della protezione più usata e del resto anche più semplice concettualmente.

Il concetto di far 'girare' un programma senza permetterne il listato o la copia spazia su campi molti vasti: non permettere la scoperta di un algoritmo importante utilizzato nel corso del programma, impedire all'utilizzatore di eliminare le protezioni inserite nel listato o la modifica stessa del listato, e in ultimo luogo, per evitare più copie di un programma specifico. Ma questo fino a un anno fa. Poi questo tipo di concetto è caduto con l'avvento sul mercato di diversi programmi che permettono di caricare in memoria e salvare su disco il programma senza mandarlo in esecuzione. Questi programmi copiatori trasferiscono il programma nella RAM sotto forma di numeri senza tradurlo in codice macchina impedendone così l'esecuzione e rendendo quindi possibile la copia.

Per togliere materialmente una protezione di questo genere, chiamata normalmente autorun (autostart), bisogna sapere principalmente quale è il suo concetto di costruzione e poi avere delle piccole conoscenze di linguaggio macchina per poter interpretare le istruzioni che costituiscono un programma di questo genere.

Come prima cosa stabilirei esattamente come si fa a riconoscere se la protezione in questione è di questo tipo. Di solito l'autorun presenta queste caratteristiche: ammettendo di prendere un programma protetto in questo senso e di caricarlo con il normale sistema di caricamento resteremo impossibilitati, una volta finito il caricamento, di intervenire in qualsiasi senso sul programma in oggetto. Sarà impossibile dare qualsiasi comando perchè il programma andrà in esecuzione senza richiedere nessun intervento all'operatore.

I metodi per effettuare questo tipo di protezione si possono dividere in tre categorie distinte. Una prima fa uso del buffer di tastiera (speciali registri nei quali vengono memorizzati dei caratteri o da tastiera o da programma) poichè esso viene svuotato alla fine di ogni esecuzione di caricamento o salvataggio, e così interpretato dal computer. La seconda, di tipo più complesso, come concetto, ma molto più semplice da realizzare, è la chiamata diretta alla routine di RUN che risiede nella memoria ROM. Il terzo tipo di autorun è profondamente diverso dagli altri due metodi e direi che è anche il più usato, perchè permette di caricare in modo sequenziale (cioè senza l'utilizzo di RUN o SYS) un altro programma strettamente legato nell'utilizzo al precedente: ad esempio i loader (caricatori) dei giochi sfruttano spesso questo sistema per impedire di salvare su supporto magnetico il programma principale che contiene il gioco vero e proprio.

Iniziamo ad analizzare il metodo che fa uso del buffer di tastiera. Possiamo realizzare la protezione sfruttando il linguaggio macchina o il BASIC.

Il buffer di tastiera è costituito da dieci

locazioni di memoria che vanno da $277 a $280 (in decimale da 631 a 640). Per bufferizzare un carattere basta fare la POKE di quella locazione di memoria con il valore numerico che gli si vuole memorizzare. Naturalmente in ogni locazione potrà essere contenuto non più di un carattere. Alla fine della immissione dei caratteri in codice ASCII bisognerà anche comunicare al computer quanti caratteri sono stati immessi; questo valore si memorizza nella locazione $C6 (#198) sempre con il metodo della POKE. Il listato per realizzare questo programmino in BASIC è mostrato in figura 1.

Il salto alla subroutine che parte dalla locazione $E544 consiste nella costruzione della pagina video. Contiene quindi anche la pulizia completa di schermo che è necessaria per evitare che il buffer di tastiera scriva su una linea già utilizzata impedendo al BASIC di eseguire il comando di RUN perchè non riconosce l'istruzione. Le linee che seguono memorizzano un valore esadecimale nell'accumulatore (registro della CPU dove si può memorizzare un dato che viene mantenuto fino all'immissione di uno nuovo) e lo trasferiscono nella locazione indicata con l'istruzione STA$xxxx, dove xxxx rappresenta la locazione di memoria in cui deve essere memorizzato il contenuto dell'accumulatore, sempre in esadecimale, poichè gli assemblatori più comunemente utilizzati comprendono solo numeri esadecimali (per come viene organizzata internamente la struttura del calcolatore).

con un normale TIM (monitor in linguaggio macchina).

Il secondo metodo consiste in sole tre istruzioni in linguaggio macchina che utilizzano la chiamata diretta alla subroutine contenuta nella ROM che produce il RUN automatico. La prima istruzione esegue un salto alla subroutine che rappresenta l'istruzione CLR del BASIC; poi salta all'indirizzo $A68E che fornisce al programma dove inizia l'area BASIC e legge il contenuto al puntatore di carattere per il byte alto e basso. In ultimo, salta alla locazione di memoria $A7AE che causa il RUN e quindi manda in esecuzione il programma. Il programma si trova in figura 4 mentre il disassemblato in figura 5.

Possiamo comunque realizzare questo programmino in linguaggio macchina ed utilizzarlo da BASIC con SYS 679 traducendolo in DATA e locandolo con il metodo di solito usato per i DATA dalla locazione 679 (locazione di partenza in decimale del programma). Il metodo utilizzato per leggere dei data possiamo vederlo nel listato in figura 2: utilizziamo una variabile dove memorizziamo la locazione di memoria di partenza ed un ciclo di FOR NEXT per incrementarla ad ogni lettura e la istruzione READ per leggere i DATA. In figura 3 c'è il disassemblaggio di questa routine.

L'ultima istruzione è un salto alla locazione $0302 dove di solito c'è l'indirizzo, che corrisponde alla routine di partenza del BASIC che è allocata a partire dalla locazione $A483. Per motivi di semplicità lascio al lettore più interessato l'analisi di questa routine che potrà trovare disassemblando il sistema operativo

Il terzo metodo, un vero e proprio programma in linguaggio macchina permette di caricare un secondo file in modo automatico rispetto al primo. Il programma è locato in una zona di memoria non interessata a nessuna variazione e più precisamente da $02C6 (#710) a $030B (#779). Lo spazio che va da $02E4 a $02F6 è riservato alla memorizzazione del nome del programma che deve essere caricato automaticamente da questa routine. In $02E4 è contenuta la lunghezza del nome e da $02E5 i

caratteri che compongono il nome.

Il programma inizia da $02C6 e termina a 02E1 e fa principalmente uso delle routine del KERNAL.

C'è una parte di programma (e più precisamente da $0300 a $030B) che modifica dei valori standard contenuti in queste locazioni per permettere di posizionare i vettori del BASIC sulla locazione voluta che nel nostro caso è $02C6, cioè da dove inizia il programma.

Il kernal è una tabella di salti (insieme di istruzioni JMP) standardizzata che serve per gestire l'ingresso, l'uscita e le routine di gestione della memoria sistema operativo. Per semplificare i programmi in linguaggio macchina si fa spesso uso di queste routine che sono allocate nell'ultima pagina della memoria ROM e più precisamente da $E000 a $FFFF.

Per usare la tabella dei salti bisogna innanzitutto fare un salto con JSR nel punto esatto della tabella ed aspettare che il kernal riporti il controllo al nostro programma con un RTS dopo che avrà svolto le sue funzioni. Molti registri possono ritornare parametri od averne bisogno per funzionare come molti altri ritornano segnali d'errore che sono molto utili per trovare dove può essere lo sbaglio che pregiudica il buon funzionamento del programma.

Nel listato mostrato in figura 6 usiamo quattro routine del kernal che adesso mi appresterò a descrivere durante la spiegazione del programma.

La routine a $FF90 controlla i messaggi di uscita del sistema e richiede che nell'accumulatore (registro A) sia memorizzato un valore a seconda che si voglia avere solo messaggi di errore come DIVISION BY ZERO o solo messaggi di controllo come PRESS PLAY ON TAPE o nessun messaggio; naturalmente il programma disabiliterà tutti i comandi del sistema in maniera tale da non permettere all'utilizzatore di sapere cosa il processore in quel momento sta facendo. La prima istruzione infatti carica nell'accumulatore il valore $00 che disabilita appunto tutti i messaggi del sistema.

Dopo viene eseguito un salto ad una subroutine creata da me che si trova a $02F7 che permette di passare al sistema i parametri del device di input tramite la routine del kernal. che si trova a $FFBA. Questa routine interessa i registri A, X, Y che rispettivamente devono contenere il numero del file logico, il numero del dispositivo e l'indirizzo secondario.

Abbiamo usato l'istruzione TAX per risparmiare un BYTE che comunque non cambia il significato richiesto dal kernal poichè questa istruzione trasferisce il contenuto dell'accumulatore nel registro X. Nel caso di utilizzo del registratore al posto di TAX bisognerà scrivere LDX #$01 che indicherà come dispositivo di input il registratore.

La prossima routine che viene utilizzata è locata a $FFBD che serve per dire al sistema dove è locato il nome del file da caricare. Bisogna caricare l'accumulatore con la lunghezza del nome del file ed i registri X e Y con l'indirizzo dove è locato il nome del file che può essere una qualsiasi locazione di memoria non influenzata da nessun parametro; l'indirizzo deve essere passato sotto standard del 6502 e cioè byte basso/byte alto.

L'ultima routine utilizzata è quella di LOAD locata a $FFD5; bisogna impostare l'accumulatore a 0 per caricare ed i registri X e Y se si desidera un caricamento rilocato. Come ultima cosa daremo una istruzione JMP all'indirizzo di partenza del programma che verrà caricato, nel listato detta locazione è a 0000, sta al lettore porre i giusti valori.

Per la partenza di un programma BASIC purtroppo bisogna inserire al posto dell'istruzione JMP la routine di directcall descritta prima e poi riaggiornare i puntatori di fine BASIC per evitare che il

```
BUFFER BASIC

    10 LOAD "NOME PRG",8,1
    20 PRINT"[CLEAR]"
    30 POKE 631,82:POKE 632,85:POKE
       633,78:POKE 634,13
    40 POKE 198,4
```

```
DIRECT CALL IN DATA

    10 LOAD "NOME PRG",8,1
    20 X=679:REM  INDIRIZZO DI PARTE
       NZA
    30 FOR Y=1 TO 9
    40 READ A:POKE X+Y,A:REM  LETTUR
       A DEI DATA
    50 NEXTX,Y
    60 SYS679
    70 DATA 32,94,166,32,142,166,74,
       174,167
```

programma si scriva sopra se stesso.

Per ulteriori approfondimenti, la Guida di Riferimento del Programmatore fornisce tutti i particolari sulla gestione del kernal e le sue routine.Per quanto riguarda la costruzione di un autorun personalizzato basterà utilizzare lo stesso programma e cambiare i parametri d'interesse. Per quanto riguarda il nome bisogna inserire da $02E5 il valore numerico dei caratteri in codice esadecimale; la lunghezza del nome viene inserita sempre in codice esadecimale in $02E4.

Per le modifiche bisogna senz'altro disporre di un monitor (programma che permette di disassemblare zone di memoria desiderate) e di alcune conoscenze del linguaggio macchina.

BUFFER IN DATA

```
10 LOAD "NOME PRG",8,1
20 X=679:REM  INDIRIZZO DI PARTE
   NZA
30 FOR Y=1 TO 30
40 READ A:POKE X+Y,A:REM  LETTUR
   A DEI DATA
50 NEXTX,Y
60 SYS679
70 DATA 32,68,229,169,82,141,119
   ,2,169,85,141,120,2,169,78,14
   1,121,2,169,13,141
80 DATA 122,2,169,4,133,198,76,2
   ,3
```

```
.,  02A7 20 44 E5 JSR $E544
.,  02AA A9 52    LDA #$52
.,  02AC 8D 77 02 STA $0277
.,  02AF A9 55    LDA #$55
.,  02B1 8D 78 02 STA $0278
.,  02B4 A9 4E    LDA #$4E
.,  02B6 8D 79 02 STA $0279
.,  02B9 A9 0D    LDA #$0D
.,  02BB 8D 80 02 STA $0280
.,  02BE A9 04    LDA #$04
.,  02C0 85 C6    STA $C6
.,  02C2 6C 02 03 JMP ($0302)
.,  02C5 00       BRK

.,  02A7 20 5E A6 JSR $A65E
.,  02AA 20 8E A6 JSR $A68E
.,  02AD 4C AE A7 JMP $A7AE
.,  02B0 00       BRK

.,  02C6 A9 00    LDA #$00
.,  02C8 20 90 FF JSR $FF90
.,  02CB 20 F7 02 JSR $02F7
.,  02CE AD E4 02 LDA $02E4
.,  02D1 A2 E5    LDX #$E5
.,  02D3 A0 02    LDY #$02
.,  02D5 20 BD FF JSR $FFBD
.,  02D8 A9 00    LDA #$00
.,  02DA A2 FF    LDX #$FF
.,  02DC A0 FF    LDY #$FF
.,  02DE 20 D5 FF JSR $FFD5
.,  02E1 4C 00 00 JMP $0000
```

```
.,  02E4 EA       NOP
.,  02E5 EA       NOP
.,  02E6 EA       NOP
.,  02E7 EA       NOP
.,  02E8 EA       NOP
.,  02E9 EA       NOP
.,  02EA EA       NOP
.,  02EB EA       NOP
.,  02EC EA       NOP
.,  02ED EA       NOP
.,  02EE EA       NOP
.,  02EF EA       NOP
.,  02F0 EA       NOP
.,  02F1 EA       NOP
.,  02F2 EA       NOP
.,  02F3 EA       NOP
.,  02F4 EA       NOP
.,  02F5 EA       NOP
.,  02F6 EA       NOP
.,  02F7 A9 08    LDA #$08
.,  02F9 AA       TAX
.,  02FA A0 01    LDY #$01
.,  02FC 20 BA FF JSR $FFBA
.,  02FF 60       RTS
.,  0300 C6 02    DEC $02
.,  0302 C6 02    DEC $02
.,  0304 C6 02    DEC $02
.,  0306 C6 02    DEC $02
.,  0308 C6 02    DEC $02
.,  030A C6 02    DEC $02
.,  030C 00       BRK
```

Il resto cinese

di **Mariangela Guardione**

L'asserto (termine matematico che vuol dire che si asserisce la dimostrazione di un teorema) noto sotto il nome di "TEOREMA CINESE DEL RESTO" ha avuto un ruolo essenziale nella dimostrazione, avvenuta nel 1970, del teorema di Matyasevich (giovane matematico russo) che fornisce la soluzione al decimo dei ventitre importanti problemi irrisolti di Hilbert.

Hilbert è stato uno dei più importanti matematici della prima metà del nostro secolo.

Questo decimo problema riguarda la risoluzione di equazioni polinomiali che devono avere come coefficienti e soluzioni solo numeri interi sia positivi che negativi e per questo sono note come "EQUAZIONI DIOFANTEE" (da Diofante di Alessandria che scrisse un libro su tale argomento nel terzo secolo D.C.).

Questo problema di Hilbert si propone di descrivere una procedura con la quale sia possibile decidere se ogni equazione diofantea ammetta soluzioni. Quindi la richiesta non riguarda la ricerca di un metodo per trovare le soluzioni, ma semplicemente fornisce un procedimento atto a determinare se l'equazione considerata possegga o meno soluzioni in quanto non è cruciale la natura dell'equazione quanto quella delle soluzioni ammesse.

Per spiegare tutto ciò si passa ad esaminare un esempio pratico considerando le equazioni $x / 2 + y / 2 - 2 = 0$ e $x / 2 + y / 2 - 3 = 0$; la prima di queste, se non la si considera come equazione diofantea, ammette infinite soluzioni che possono venire rappresentate con i punti di una circonferenza nel piano delimitato dagli assi x e y e il cui centro è posto nell'origine degli assi e il raggio misura $\sqrt{2}$.

Tuttavia se la si considera come un'equazione diofantea essa ammette solo quattro soluzioni intere che sono: $x = 1, y = -1$; $x = 1, y = 1$; $x = 1, y = -1$; $x = 1, y = -1$.

Mentre la seconda equazione avrà, sempre se viene considerata come equazione usuale, un numero infinito di soluzioni anch'esse rappresentabili con punti di una circonferenza con centro nell'origine degli assi e raggio $\sqrt{3}$, ma non avrà invece soluzioni nel caso la si consideri equazione diofantea in quanto non esiste nessun punto di tale curva che abbia le coordinate simultaneamente uguali ad un numero intero.

Una famiglia di equazioni diofantee è quindi rappresentabile come $x / n + y / n = z / n$ con n uguale a 2,3,4 o a qualsiasi intero positivo.

Nel caso in cui si abbia $n = 2$ la soluzione dell'equazione $x / 2 + y / 2 = z / 2$ è data dalla lunghezza dei lati di un triangolo rettangolo ed è nota come teorema di Pitagora. Se $n > = 3$ l'equazione che si ottiene è nota sotto il nome di "EQUAZIONE DI FERMAT" e la sua dimostrazione (se mai fu ottenuta dal matematico francese) non è mai stata scoperta.

Tutti gli esempi che sono stati illustrati hanno dimostrato come le equazioni diofantee siano facili da scriversi, ma molto complesse da risolversi in quanto necessitano di condizioni molto esclusive riguardo il tipo di numeri da accettare come loro soluzioni. La domanda che può sorgere a questo punto è:

perché è difficile trovare un metodo d'indagine matematica del tipo di quello richiesto da Hilbert?

Una risposta immediata sarebbe quella di sperimentare tutti i possibili valori delle incognite fino a trovare una soluzione.

Questo procedimento, ammesso di poter utilizzare un computer, non potrebbe essere già più utilizzato per l'equazione considerata preceden-

temente x / 2 + y / 2—3 = 0 in quanto non riusciremmo mai a stabilire se la coppia successiva potrebbe o meno essere la soluzione desiderata. È per questo motivo che una tale dimostrazione richiede considerazioni matematiche di nuovo tipo.

Un meccanismo in grado di ottenere un procedimento del tipo suggerito da Hilbert dovrebbe essere rappresentato da una macchina che può essere chiamata "macchina di Hilbert", la quale dovrebbe accettare in ingresso i coefficienti di una qualunque equazione diofantea e come uscita dovrebbe accendersi una luce verde nel caso in cui l'equazione ammetta una soluzione mentre s'accende una luce rossa in caso contrario. (Fig. 1)

A questo punto il problema che sorge è se sia possibile costruire una macchina a luce verde e rossa che si fermi sempre dopo un numero finito di passi fornendo sempre una risposta positiva o negativa. Quello che è stato dimostrato dal giovane matematico russo Matyasevich è che non potrà mai venir compilato un programma né costruita una macchina che esegua ciò che è stato postulato nel decimo problema di Hilbert.

Per spiegare l'asserto precedente bisogna fare alcune considerazioni che riguardano la computabilità. Per questo si supponga che S indichi l'insieme dei numeri interi: sarà quindi elencabile solo se potrà essere costruita una macchina a luce verde che accetti come ingresso un qualunque numero intero e come uscita s'accenda la luce verde dopo un numero finito di passi se e solo se il numero intero in ingresso appartiene ad S mentre s'accenda la luce rossa se l'intero non appartiene a detto insieme.

Per una migliore comprensione di questo concetto si supponga che S indichi il complemento di S, cioè sia l'insieme di tutti gli interi che non appartengono ad S. Se ad esempio S è l'insieme dei numeri interi pari, S è l'insieme di quelli dispari.

Si può dimostrare che se S è computabile allora sia S che S saranno elencabili. Questo vuol dire che se esiste per S una macchina a luce verde e rossa allora esiste una macchina a luce verde per S e una a luce verde per S. (Fig. 2) È vera anche l'affermazione inversa: se S e S sono elencabili, allora S è computabile; in modo analogo si può anche dire che se esiste una macchina a luce verde sia per S che per S, allora può essere costruita per S una macchina a luce verde e rossa. Tutto questo si può ottenere sostituendo nella macchina a luce verde per S la lampadina verde con una rossa e quindi si colleghino in parallelo le due macchine in modo che uno stesso ingresso valga simultaneamente per entrambe in modo da ottenere una macchina a luce verde e rossa. (Fig. 3,4 e 5).

Da queste considerazioni si può ricavare la soluzione ad uno dei problemi più cruciali della teoria della computabilità che gioca un ruolo molto importante nella soluzione del decimo problema di Hilbert: cioè esiste un insieme K elencabile, ma non computabile, che in termini di una macchina di Hilbert, vuol dire che esiste una macchina a luce verde per K ma non è possibile costruire una macchina a luce verde per il complemento di K, cioè per K.

A questo riguardo, come si può essere sicuri che per K non esista alcuna macchina a luce verde? Per dimostrare questo si postula che esista una tale macchina corredata di un libretto di istruzioni e che dal momento che K è il complemento di K, si avrà che comunque si fornisca un numero in ingresso, ad esempio 297, a tale macchina, essa si accenderà se e solo se in uscita M297 non si accende per 297; se questo si verificasse tutto ciò starebbe a significare che l'intero 297 appartiene a K e non a K. Quindi la macchina per K non è certamente M297, ma per la stessa ragione essa non coinciderà con nessuna Mn per qualunque altro valore di n, in quanto lo stesso ragionamento potrebbe venire applicato anche per ogni altro numero, dimostrando che nell'elenco di tutti i libretti d'istruzione compare in nessun punto una macchina a luce verde per K. Si ricava quindi che non può esistere alcuna macchina a luce verde per K cioè K non è elencabile. (Fig. 6 e 7).

Quindi si può affermare che non esisterà mai un algoritmo per separare K da K. Tutto questo venne utilizzato da Matyasevich per dimostrare che ogni insieme elencabile può essere fatto corrispondere ad un'equazione diofantea; cioè se S è un'insieme elencabile esisterà allora un polinomio P con coefficienti interi e con variabili x,y1,y2,.....yn che corrisponde a tale insieme e che verrà indicato con P(x,y1,y2,....,yn). Dal risultato ottenuto dal matematico russo si può otte-

nere un'equazione diofantea P (x,y1,y2,...,yn) = 0 associata a tale insieme e se fosse possibile la costruzione di una macchina a luce verde e rossa per verificare se esistono o meno soluzioni, allora per ogni intero x si potrà stabilire se esistono o meno degli interi y1,y2,...,yn per cui l'equazione precedente abbia soluzioni.

Per far questo si arriverà anche a determinare se x appartiene o meno all'insieme k.

Ma precedentemente si è già dimostrato che K non è computabile e quindi l'unica conclusione che permette la soluzione di questo dilemma è quella di ammettere che non esiste alcuna macchina di Hilbert e quindi ottenere il risultato che il decimo problema di Hilbert non è risolubile.

Come è stato detto all'inizio di questo articolo la dimostrazione di Matyasevich si basa sul TEOREMA CINESE DEL RESTO che garantisce l'esistenza del più piccolo numero n che diviso per divisori dati dia resti predeterminati che devono essere interi positivi qualsiasi. Unica ipotesi richiesta affinché si possa applicare il resto cinese è che nessuna coppia di divisorì utilizzata abbia fattori comuni eccetto l'unità.

Per illustrare il suddetto teorema, prima di passare alla spiegazione del programma sul resto cinese qui allegato, si è voluto dare un esempio numerico. Si supponga di voler trovare un numero N che diviso rispettivamente per 10,3,7,11 dia come resti 4,2,3,1.

Per ottenere il numero desiderato si procede nel seguente modo: si indica con x il numero cercato e si scrive quindi

RES (x/10) = 4 RES (x/7) = 3

RES (x/3) = 2 RES (x/11) = 1

a questo punto per determinare x si devono risolvere quattro equazioni ausiliari in cui compaiono quattro nuove incognite che indicheremo con y1,y2,y3,y4. I nuovi numeratori saranno dati da un coefficiente ottenuto moltiplicando tre dei divisori per una delle nostre nuove incognite mentre il denominatore sarà il quarto divisore; per meglio illustrare questo procedimento si osserva che il primo numeratore sarà dato da 231y1 ottenuto nel seguente modo: 3*7*11 = 231, analogamente si opera per gli altri e si ottiene:

RES (231y1/10) = 4 RES (330y3/7) = 3

RES (770y2/3) = 2 RES (210y4/11) = 1

a questo punto per trovare le nuove incognite si procede iterativamente sostituendo ad y1,y2,y3,y4 dei valori ad incrementarsi partendo da 1 sino a che il prodotto fra essi e i loro coefficienti diviso il denominatore dia il resto richiesto.

In termini numerici questo vuol dire che nel primo caso si inizia con y1 = 1 e si ottiene 231*1/10 = 1 che non rappresenta il resto richiesto e quindi si prova con y1 = 2,3,4. Per quest'ultimo valore di y1 si ottiene 231*4/10 = 4 che rappresenta il resto voluto; quindi il valore cercato è y1 = 4.

Questa operazione viene ripetuta per tutte le altre incognite. Il numero cercato x si ottiene quindi nel seguente modo:

x = (231*4) + (770*1) + (330*3) + (201*1) = 2894

che è quindi un possibile valore di x. Per trovare il più piccolo valore di x si deve sottrarre al valore trovato il prodotto dei quattro divisori cioè:

2894—(10*3*7*11) = 584

che rappresenta la più piccola soluzione del problema.

Per concludere si vuole illustrare come è stato strutturato il programma per il COMMODORE 64.

Nella prima parte si esegue il test per verificare che i quattro divisori immessi non abbiano termini in comune; per far questo è stato utilizzato l'algoritmo del M.C.D. trattato nell'articolo "SCOMPOSIZIONE DI UN NUMERO DI FATTORI PRIMI, CALCOLO DEL M.C.D. E DEL m.c.m FRA DUE NUMERI" e quindi si passa al confronto dei divisori fra loro.

La seconda parte del programma riguarda il calcolo vero e proprio del resto cinese nella sequenza che è stata illustrata nell'esempio numerico precedentemente descritto.

A conclusione di questo articolo viene riportato il listato del programma in linguaggio BASIC del "RESTO CINESE".

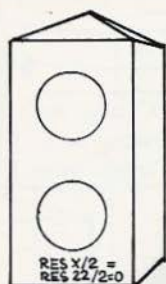



FIG.1
RES X/2 =
RES 22/2=0
22 E' UN ELEMENTO
DI S?
RES X/2 =
RES 23/2 =1
23 E' UN ELEMENTO
DI S?
E' POSSIBILE TRASFORMARE
UNA MACCHINA A LUCE VERDE
E ROSSA IN UNA MACCHINA
A LUCE VERDE PER S
PIU' UNA MACCHINA A
LUCE VERDE PER IL
COMPLEMENTO DI S
S
SI, POSSO
DIMOSTRARLO
COLLEGANDO...
S
... DIVERSAMENTE
LE LAMPADINE
S
FIG. 2

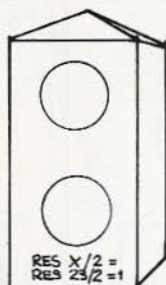
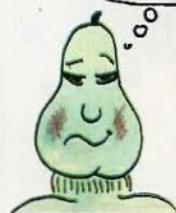
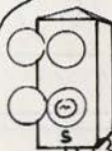
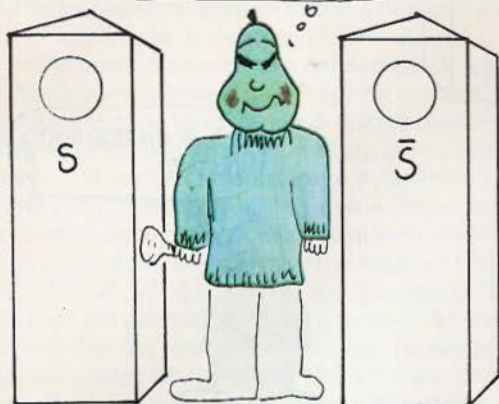
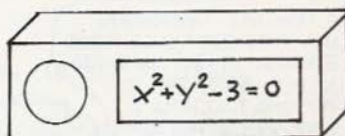
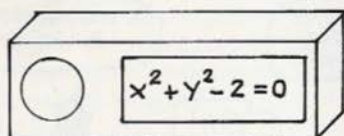
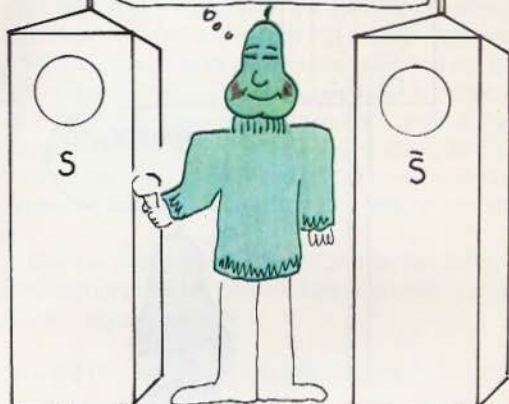
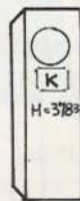
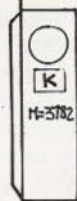
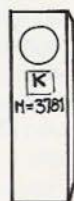
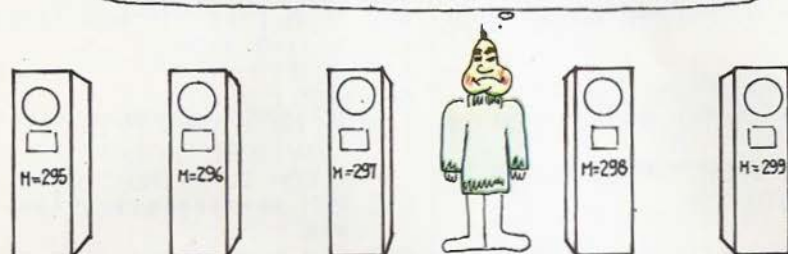



FIG. 3
POSSO INVECE UTILIZZARE UNA MACCHINA A LUCE VERDE PER S PIU' UNA MACCHINA A LUCE VERDE PER IL COMPLEMENTO DI S...
S
S̄
FIG 4
E UNA LAMPADINA ROSSA PER COSTRUIRE UNA MACCHINA A LUCE VERDE E ROSSA PER S
S
S̄
$x^2+y^2-2=0$
$x^2+y^2-3=0$
FIG 5
SCEGLIENDO UN NUMERO QUALUNQUE (AD ESEMPIO 3781) K E' DEFINITO COME L'INSIEME DEI NUMERI TALI CHE L'N-ESIMA MACCHINA A LUCE VERDE SI ACCENDE SE LE SI PONE IN INGRESSO IL NUMERO N
K
N=3779
K
N=3780
K
N=3781
K
N=3782
K
N=3783
FIG 6

```
100 REM *************************
    **
105 REM *
    *
110 REM *  RESTO CINESE
    *
115 REM *        DI
    *
120 REM *  MARIANGELA GUARDIONE
    *
125 REM *
    *
130 REM *************************
    **
135 DIM  A(50),B(50),C(50),D(50),
    R(4)
140 REM *************************
    **
145 REM * RINVIO SUBROUTINE INPUT
    *
150 REM *************************
    **
155 GOSUB  320
160 REM *************************
    **
165 REM * INIZIO CALCOLO DEL
    *
170 REM * RESTO CINESE
    *
175 REM *************************
    **
180 NUM=B(0)*C(0)*D(0):DEN=A(0):R
    R=R(1):GOSUB  275:Y1=I
185 Y1=Y1*NUM
190 NUM=A(0)*C(0)*D(0):DEN=B(0):R
    R=R(2):GOSUB  275:Y2=I
195 Y2=Y2*NUM
200 NUM=A(0)*B(0)*D(0):DEN=C(0):R
    R=R(3):GOSUB 275:Y3=I
205 Y3=Y3*NUM
210 NUM=A(0)*B(0)*C(0):DEN=D(0):R
    R=R(4):GOSUB 275:Y4=I
215 Y4=Y4*NUM
220 REM ****************
225 REM * MINIMO NUMERO *
230 REM ****************
235 N=Y1+Y2+Y3+Y4-A(0)*B(0)*C(0)*
    D(0)
240 PRINT N;"[RVS]E' IL NUMERO CE
    RCATO[RVOFF]"
245 INPUT "VUOI CONTINUARE(S/N)";
    S$
250 IF S$="S" THEN 155
255 END
260 REM *************************
    **
265 REM * SUBROUTINE RESTO CINESE
    *
270 REM *************************
    **
275 I=1
280 RS=INT(.2+((NUM*I/DEN)-INT(NU
    M*I/DEN))*DEN)
285 IF RS<>RR THEN RETURN
290 I=I+1:GOTO 280
295 REM *************************
    ***
300 REM * INIZIO CONFRONTO DIV.
    *
305 REM * SCOMPOSIZIONE IN FATTOR
    I *
310 REM * PRIMO DIVISORE
```

```
     *
315 REM ************************
    ***
320 INPUT "IMPOSTARE IL PRIMO DIV
    ISORE";A(0):FLAG=0:KO=1:N=A(0
    )
325 I=2
330 IF I>=N THEN 365
335 IF ((N/I)-INT(N/I))=0 THEN 35
    0
340 IF I>N/2 THEN I=N:GOTO 365
345 I=I+1:GOTO 330
350 FLAG=1
355 N=N/I:A(KO)=I:KO=KO+1:IF N=1
    THEN 375
360 GOTO 325
365 IF FLAG=0 THEN A(1)=N:KO=2:GO
    TO 375
370 GOTO 350
375 TN=KO-1
380 REM ************************
    ***
385 REM * SCOMPOSIZIONE IN FATTOR
    I *
390 REM * SECONDO DIVISORE
     *
395 REM ************************
    ***
400 INPUT "IMPOSTARE IL SECONDO D
    IVISORE";B(0):FLAG=0:KO=1:M=B
    (0)
405 I=2
410 IF I>=M THEN 445
415 IF ((M/I)-INT(M/I))=0 THEN 43
    0
420 IF I>M/2 THEN I=M:GOTO 445
425 I=I+1:GOTO 410
430 FLAG=1
435 M=M/I:B(KO)=I:KO=KO+1:IF M=1
    THEN 475
440 GOTO 405
445 IF FLAG=0 THEN B(1)=M:KO=2:GO
    TO 475
450 GOTO 400
455 REM ************************
    ***
460 REM * CONFRONTO DI FATTORI FR
    A *
465 REM * PRIMO E SECONDO DIVISOR
    E *
470 REM ************************
    ***
475 TM=KO-1:MD=1
480 FOR I=1 TO TN
485 FOR J=1 TO TM
490 IF A(I)=B(J) THEN MD=MD*B(J):
    B(J)=0:J=TM
495 NEXTJ
500 NEXTI
505 IF MD>1 THEN PRINT"IL SECONDO
     DIVISORE HA FATTORI IN COMUN
    E CON IL PRIMO!":GOTO 400
510 REM ************************
    ***
515 REM * SCOMPOSIZIONE IN FATTOR
    I *
520 REM * TERZO DIVISORE
     *
525 REM ************************
    ***
530 INPUT "IMPOSTARE IL TERZO DIV
    ISORE";C(0):FLAG=0:KO=1:P=C(0
    )
535 I=2
540 IF I>=P THEN 575
545 IF ((P/I)-INT(P/I))=0 THEN 56
    0
550 IF I>P/2 THEN I=P:GOTO 575
555 I=I+1:GOTO 540
560 FLAG=1
565 P=P/I:C(KO)=I:KO=KO+1:IF P=1
    THEN 605
570 GOTO 535
575 IF FLAG=0 THEN C(1)=P:KO=2:GO
    TO 605
580 GOTO 560
585 REM ************************
    ***
590 REM * CONFRONTO DI FATTORI FR
    A *
595 REM * PRIMO E TERZO DIVISORE
     *
600 REM ************************
    ***
605 TP=KO-1:MD=1
610 FOR I=1 TO TN
615 FOR J=1 TO TP
620 IF A(I)=C(J) THEN MD=MD*C(J):
    C(J)=0:J=TP
625 NEXTJ
630 NEXTI
635 IF MD>1 THEN PRINT"IL TERZO D
```

```
    IVISORE HA FATTORI IN COMUNE
    CON IL PRIMO!":GOTO 530
640 REM ************************
    ***
645 REM * CONFRONTO DI FATTORI FR
    A *
650 REM * SECONDO E TERZO DIVISOR
    E *
655 REM ************************
    ***
660 FOR I=1 TO TM
665 FOR J=1 TO TP
670 IF B(I)=C(J) THEN MD=MD*C(J):
    C(J)=0:J=TP
675 NEXTJ
680 NEXTI
685 IF MD>1 THEN PRINT"IL TERZO D
    IVISORE HA FATTORI IN COMUNE
    CON IL SECONDO!":GOTO 530
690 REM ************************
    ***
695 REM * SCOMPOSIZIONE IN FATTOR
    I *
700 REM * QUARTO DIVISORE
      *
705 REM ************************
    ***
710 INPUT "IMPOSTARE IL QUARTO DI
    VISORE";D(0):FLAG=0:KO=1:Q=D(
    0)
715 I=2
720 IF I>=Q THEN 755
725 IF ((Q/I)-INT(Q/I))=0 THEN 74
    0
730 IF I>Q/2 THEN I=Q:GOTO 755
735 I=I+1:GOTO 720
740 FLAG=1
745 Q=Q/I:C(KO)=I:KO=KO+1:IF Q=1
    THEN 785
750 GOTO 715
755 IF FLAG=0 THEN D(I)=Q:KO=2:GO
    TO 785
760 GOTO 740
765 REM ************************
    ***
770 REM * CONFRONTO DI FATTORI FR
    A *
775 REM * PRIMO E QUARTO DIVISORE
      *
780 REM ************************
    ***
785 TQ=KO-1:MD=1
790 FOR I=1 TO TN
795 FOR J=1 TO TQ
800 IF A(I)=D(J) THEN MD=MD*D(J):
    D(J)=0:J=TQ
805 NEXTJ
810 NEXTI
815 IF MD>1 THEN PRINT"IL QUARTO
    DIVISORE HA FATTORI IN COMUNE
     CON IL PRIMO!":GOTO 710
820 REM ************************
    ***
825 REM * CONFRONTO DI FATTORI FR
    A *
830 REM *]SECONDO E QUARTO DIVISO
    RE*
835 REM ************************
    ***
840 FOR I=1 TO TM
845 FOR J=1 TO TQ
850 IF B(I)=D(J) THEN MD=MD*D(J):
    D(J)=0:J=TQ
355 NEXTJ
860 NEXTI
865 IF MD>1 THEN PRINT"IL QUARTO
    DIVISORE HA FATTORI IN COMUNE
     CON IL SECONDO!":GOTO 710
870 REM ************************
    ***
875 REM * CONFRONTO DI FATTORI FR
    A *
880 REM * TERZO E QUARTO DIVISORE
      *
885 REM ************************
    ***
890 FOR I=1 TO TP
895 FOR J=1 TO TQ
900 IF C(I)=D(J) THEN MD=MD*D(J):
    D(J)=0:J=TQ
905 NEXTJ
910 NEXTI
915 IF MD>1 THEN PRINT"IL QUARTO
    DIVISORE HA FATTORI IN COMUNE
     CON IL TERZO!":GOTO 710
920 REM ********************
925 REM * INPUT DEI RESTI   *
930 REM ********************
935 FOR I=1 TO 4:PRINT"IMPOSTARE
    IL";I;"O RESTO":INPUT R(I)
940 NEXTI
945 RETURN
```

Vendo Magic Desk a L. 10.000. Vendo traduzione italiana riveduta e corretta del Manuale d'uso. *(Giuliano Peritore - Via Amaseno, 6 - 04100 Latina).*

Vendo/Cambio Atari VCS 2600 6 mesi + Supercharger + 7 cassette. Cambio (eventuale conguaglio) con C64. *(Stefano Saia - Via S. Gorizia, 29 - 15100 Alessandria - Tel. 0131-51823).*

Vendo videogioco Atari con dodici cartucce gioco al miglior offerente. *(Edoardo Zappa - Via Stendhal, 19 - 20144 Milano - Tel. 02-4227764 - ore pasti serali).*

Vendo Commodore Vic 20 + vari videogiochi e vari programmi. Tutto a L. 250.000= *(Cesare Susanna - Via S. Ilaria, 2 - 00199 Roma - Tel. 06-8388577).*

Vendo tastiera Vic 20 + 16K espansione + The Vic 20 Games Book + Cartridge: Alien + Alla scoperta del Vic + 1 cassetta giochi a L. 370.000. Cassette giochi a parte L. 8.000. *(Walter Calioni - Via dei Mille, 26 - 00185 Roma - Tel. 06-4956604).*

Vendo per Vic 20 nuovissimo cartridge Turbotape per usare il registratore alla velocità di un disco con connettore per eventuali espansioni di memoria, L. 38.000 comprese spese postali. Vicmon L. 29.000. Programmer's Aid L. 29.000. Motherboard L. 35.000. *(Gianni Moritz Bozzi - Via Savona 16/s - 20099 Sesto San Giovanni MI - Tel. 02-2407825).*

Vendo Vic 20 in ottimo stato con due cassette di giochi, numerosi listati a L. 200.000 *(Olderico Caviglia - Via D. Carbone, 4 - 15050 Villalvernia - Alessandria - Tel. 0131-83150 ore pasti).*

Vendo Vic 20 con registratore più serie programmi. *(Roberto Covanti - Via Serralonga, 52 - 60044 Fabriano - Tel. 0732-3915 ore pasti).*

Vendo Commodore 64, registratore dedicato, 2 cassette videogioco ed altre 3 con programmi vari. Nuovo maggio '84, usato pochissimo a L. 650.000= *(Alfonso Cantarella - Via Mercato, 2 - 84015 Nocera Superiore - Tel. 081-934350 ore: 13-14 22-23).*

Vendo corso di basic per ogni Computer, manuale dettagliatissimo con tanti esempi e programmi e corso generale per l'utilizzo e l'impiego del computer a L. 18.000 per manuale. *(Guido Ghidetti - Via Campioni, 9 - 43040 Felegara (Pr)).*

Vendo Vic 20 con cavi e alimentatori, manuali + Registratore Commodore C2N + Superexpander + Cartridge Solar System + televisorino b/n 16 pollici + coprisistema + cassette di software didattico, gestionale di utility, di grafica tridimensionale, qualche gioco + libro «Grafica per Vic» per la S. Expander. Tutto all'incredibile prezzo di L. 490.000= non trattabili. *(Andrea Buffagni - Via G. Peano, 6 - 41100 Modena - Tel. 059-354424 ore pasti).*

Vendo per Commodore C64 disco e manuale in italiano per lo sblocco di qualsiasi programma protetto e 2 Backup speciali prezzo L. 65.000 tutto compreso. *(Leonardo Landini - Via Corcos, 5 - 50100 Firenze).*

Vendo Computer Vic 20 + registratore dedicato + corso basic + joystick + 4 cartridge (Vic Avenger/Raid on Fort Knox/Mission impossible Adventure/ Super Alien) + 120 giochi su nastro. Tutto completo di manuale a L. 400.000= *(Eugenio Afelici - Via A. Moro trav. Neri, 26 - 89100 Reggio Calabria - Tel. 590588 ore 21-23 14-15).*

Vendo Commodore 64 + 120 programmi vari a prezzo modico. *(Giuseppe Borracci - Via Mameli, 15 - 33100 Udine - Tel. 0432-291665, ore 13-13,30 - 21-22).*

Vendo causa regalo nuovo computer, Sinclair ZX81, memoria 16K, manuale in italiano, cassetta gioco scacchi originale. Il tutto usato pochissimo a sole L. 200.000= *(Francesco Monaco - Via Forlì, 74/1 - 20090 Sesto San Giovanni MI - Tel. 02-2421334 ore 19,30-20,30).*

Vendo Vic 20 + 4 registratore + 4 cartucce di giochi (con imballo originale) perchè ne ho avuti due in regalo in giugno, a L. 280.000, mai usato. *(Sergio Santoro - Via Agnoletti, 20 - 50047 Prato - Tel. 0574-465904 alla mattina).*

Vendo Vic 20 completo di manuale istruzioni + interessantissimo libro di grafica per il Vic. L. 170.000 con programmi in omaggio. *(Luca Mosini - Via E. Chiesa, 55 - 00139 Roma - Tel. 06-8107685 ore 12-14).*

Vendo registratore Phonemark nuovo per il Vic 20 e Commodore 64 a L. 75.000. *(Franceschini - Via Meloria, 7 - 20148 Milano - Tel. 02-3272886 ore 9-12).*

Vendo Vic 20 + registratore + espansione di memoria 16 K + 4 videogiochi, ancora imballato a prezzo eccezionale: L. 400.000= *(Raffaele Addeo - Via Trento, 1/C - 20060 Cassina de'Pecchi MI - Tel. 02-9528793 ore serali).*

Vendo a L. 500.000 floppy disk 1541, non trattabili perchè nuovo (è in garanzia). Vendo inoltre moltissimi nuovi games (jump-man, Olimpiadi, Basket, Killer Watter, ecc.) ed utility tra cui Condominio, G-Pascal, Forth, Doudle (nuovo superprogramma per disegnare, occupa un disco intero). 5.5 Fast copy, prezzi da regalo. *(Ornella Vergotti - Via Castello, 6574 - 30122 Venezia).*

Vendo interfaccia per collegare al Vic 20 od al Commodore 64 qualsiasi registratore. Nuovissima L. 20.000 tratt. Vendo/Scambio programmi molto buoni a prezzi bassissimi, utility, giochi. *(Roberto Fusco - Via Crocifisso 1/G - 01100 Viterbo - Tel. 0761-220383).*

Vendo Vic 20 + registratore C2N + Super expander +8K Ram + Bi-slot + Prog. Reference + Vic Revelead + 2 cassette programmi a 1; 550.000 trattabili. Preferibilmente tratto di persona. *(Domenico Pozzetto - V.le Virgilio, 48 - 34170 Gorizia - Tel. 83775).*

Vendo Vic 20 completo di manuale di istruzioni «Impariamo il basic con il Vic/ CBM - Guida al Personal Vic 20» con mappe di memoria + una cassetta di giochi e utility tutto a L. 150.000. Vendo causa passaggio a sistema superiore. *(Salvatore Sassi - Via San Lazzaro - cool. Alfa - 85170 Isernia - Tel. 0865-59112-51604 ore: 13,30-15,30 dopo le 22).*

Vendo Vic 20 + espansione 16K + Mother Board + Tool Kit + Cartuccia Gorf+Monitor in linguaggio macchina + registratore Commodore + libro «Impariamo a programmare con il Vic e BM» + cassette originali come Skramble, Pixel Power, Load'n Run, programmi, ecc. più varie copie di giochi come: Jet Pac, Catch Snatcha, Pades and Mutant, Escape M.C.P. e molti altri. In tutto sono più di 200 programmi, ho anche moltissime utility per la casa e il negozio. Sono a disposizione di chiunque voglia provare il tutto prima di un eventuale acquisto. Vendo tutto il blocco per L. 550.000= trattabili; valore commerciale più di L. 900.000= PS. Offro ed esigo massima serietà. Telefonare di pomeriggio fino alle ore 17,00 o la sera dopo le 21. *(Mimmo Moraca - Piazza Pitagora, 32 - 88074 Crotone - Tel. 0962-25307).*

Vendo istruzioni Vic 20 e istruzioni floppy disk drive in italiano, sempre in italiano vendo manuali Simon's Basic e Easy Script, Tavoletta grafica per CBM 64. Prezzi veramente ottimi. *(chiedere di Paolo al numero telefonico: 071-895579).*

Vendo a prezzo eccezionale: solo L. 50.000, TV Sport collegabile a televisione a colori o in b/n con 6 giochi: calcio, squash, training, tennis, tiro a segno con pistola e fucile inclusi. Diverse possibilità di gioco *(Luca Ardenghi - Via Marco Polo, 29 - 61032 Fano - Tel. 0721-861068).*

Se vuoi abbonarti

Registrate il mio abbonamento annuale a Commodore.

☐ Ho versato oggi stesso il canone di Lire 25.000 a mezzo c/c postale n° 31532203 intestato a: Commodore Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

☐ Accludo assegno per lire 25.000 banca .... n° .... a favore di Commodore Systems Editoriale

Il mio computer è: VIC 20 ☐, C 64 ☐, altro (specificare) ....

Ho ☐ / non ho ☐ la stampante, ma voglio ☐ comprarla.

Preferisco programmi di gioco ☐, didattici ☐, d'utilità ☐, altro ....

Nome .... Cognome ....

Via .... n° .... CAP. ☐☐☐☐☐ Città ....

Tel. ....

Se vuoi collaborare

Registratemi fra i collaboratori regolari di Commodore.

A titolo di prova vi invio un articolo e la cassetta col programma "............ ...." di cui vi garantisco l'assoluta originalità autorizzandovene la pubblicazione.

☐ Scrivetemi all'indirizzo sottoindicato ....

Nome ....

Via .... N° ....

Tel. .... CAP .... Città ....

Se vuoi un consiglio o consigliarci

HELP ....

....

....

....

....

Nome ....

Via .... n° .... CAP. ☐☐☐☐☐ Città ....

Tel. .... Orario ....

Il mio computer è configurato:

			Sono in possesso	No	Ho intenzione di acquistare
Vic 20 ☐	espanso a ....K		☐	☐	☐
C 64 ☐			☐	☐	☐
Floppy ☐	quale: 1541 ☐	altro	☐	☐	☐
Stampante ☐	quale: MPS801 ☐	altro	☐	☐	☐
Plotter ☐	quale: 1520 ☐	altro	☐	☐	☐
Registratore ☐	quale: 1530 ☐	altro	☐	☐	☐

Televisore ☐, TV-Monitor ☐, Monitor ☐, Colore ☐, B/N ☐

Nome .... Cognome ....

Via .... n° .... CAP. ☐☐☐☐☐ Città ....

Tel. ....

Se vuoi vendere o comprare

Vendo ☐ Compro ☐ ....

....

....

....

....

Nome ....

Via .... n° .... CAP. ☐☐☐☐☐ Città ....

Tel. .... Orario ....

Da inviare in busta chiusa a:

Spett.le rivista
Commodore
Systems Editoriale

v.le Famagosta, 75
20142 Milano

Si, voglio abbonarmi

Da inviare in busta chiusa a:

Spett.le rivista
Commodore
Systems Editoriale

v.le Famagosta, 75
20142 Milano

Si, voglio collaborare

Da inviare in busta chiusa a:

Spett.le rivista
Commodore
Systems Editoriale

v.le Famagosta, 75
20142 Milano

Si, chiedo consiglio

Da inviare in busta chiusa a:

Spett.le rivista
Commodore
Systems Editoriale

v.le Famagosta, 75
20142 Milano

Si, voglio votare

Da inviare in busta chiusa a:

Spett.le rivista
Commodore
Systems Editoriale

v.le Famagosta, 75
20142 Milano

Si vendo/ compro